Eur. Phys. J. B **66**, 409–418 (2008)
DOI: 10.1140/epjb/e2008-00425-1

THE EUROPEAN PHYSICAL JOURNAL B

Modularity-maximizing graph communities via mathematical programming

G. Agarwal[1,2] and D. Kempe[1,a]

[1] Computer Science Department, University of Southern California, CA 90089-0781, Los Angeles, USA
[2] Google Inc., Hyderabad, India

Received 20 April 2008 / Received in final form 8 September 2008
Published online 27 November 2008 – © EDP Sciences, Società Italiana di Fisica, Springer-Verlag 2008

Abstract. In many networks, it is of great interest to identify *communities*, unusually densely knit groups of individuals. Such communities often shed light on the function of the networks or underlying properties of the individuals. Recently, Newman suggested *modularity* as a natural measure of the quality of a network partitioning into communities. Since then, various algorithms have been proposed for (approximately) maximizing the modularity of the partitioning determined. In this paper, we introduce the technique of rounding mathematical programs to the problem of modularity maximization, presenting two novel algorithms. More specifically, the algorithms round solutions to linear and vector programs. Importantly, the linear programing algorithm comes with an a posteriori approximation guarantee: by comparing the solution quality to the fractional solution of the linear program, a bound on the available "room for improvement" can be obtained. The vector programming algorithm provides a similar bound for the best partition into *two* communities. We evaluate both algorithms using experiments on several standard test cases for network partitioning algorithms, and find that they perform comparably or better than past algorithms, while being more efficient than exhaustive techniques.

PACS. 89.75.Hc Networks and genealogical trees – 05.10.-a Computational methods in statistical physics and nonlinear dynamics – 02.10.Ox Combinatorics; graph theory – 87.23.Ge Dynamics of social systems

1 Introduction

Many naturally occurring systems of interacting entities can be conveniently described using the notion of networks. *Networks* (or *graphs*) consist of *nodes* (or *vertices*) and *edges* between them [1]. For example, *social networks* [2,3] describe individuals and their interactions, such as friendships, work relationships, sexual contacts, etc. Hyperlinked text, such as the World Wide Web, consists of pages and their linking patterns [4]. Metabolic networks model enzymes and metabolites with their reactions [5].

In analyzing and understanding such networks, it is frequently extremely useful to identify *communities*, which are informally defined as "unusually densely connected sets of nodes". Among the benefits of identifying community structure are the following:

1. Frequently, the nodes in a densely knit community share a salient real-world property. For social networks, this could be a common interest or location; for web pages, a common topic or language; and for biological networks, a common function. Thus, by analyzing structural features of a network, one can infer semantic attributes.
2. By identifying communities, one can study the communities individually. Different communities often exhibit significantly different properties, making a global analysis of the network inappropriate. Instead, a more detailed analysis of individual communities leads to more meaningful insights, for instance into the roles of individuals.
3. Conversely, each community can be compressed into a single "meta-node", permitting an analysis of the network at a coarser level, and a focus on higher-level structure. This approach can also be useful in visualizing an otherwise too large or complex network.

For a much more detailed discussion of these and other motivations, see for instance [6]. Due to the great importance of identifying community structure in graphs, there has been a large amount of work in computer science, physics, economics, and sociology (for some examples, see [6–10]). At a very high level, one can identify two lines of work. In one line [7,11], dense communities

[a] e-mail: dkempe@usc.edu

are identified one at a time, which allows vertices to be part of multiple communities. Depending on the context, this may or may not be desirable. Often, the communities identified will correspond to some notion of "dense subgraphs" [7,11–13].

An alternative is to seek a *partition* of the graph into disjoint communities, i.e., into sets such that each node belongs to exactly one set. This approach is preferable when a "global view" of the network is desired, and is the one discussed in the present work. It is closely related to the problem of *clustering*; indeed, "graph clustering", "partitioning", and "community identification" are often, including here, used interchangeably.

Many approaches have been proposed for finding such partitions, based on spectral properties, flows, edge agglomeration, and many others (for a detailed overview and comparison, see [6]). The approaches differ in whether or not a hierarchical partition (recursively subdividing communities into sub-communities) is sought, whether the number of communities or their size is pre-specified by the user or decided by the algorithm, as well as other parameters. For a survey, see [9].

A particularly natural approach was proposed by Newman and Girvan [14,15]. Newman [15] proposes to find a community partition maximizing a measure termed *modularity*. The modularity of a given clustering is the number of edges inside clusters (as opposed to crossing between clusters), minus the expected number of such edges if the graph were random conditioned on its degree distribution [14] . Subsequent work by Newman et al. and others has shown empirically that modularity-maximizing clusterings often identify interesting community structure in real networks, and focused on different heuristics for obtaining such clusterings [6,10,14–18]. For a detailed overview and comparison of many of the proposed heuristics for modularity maximization, see [19].

Remark 1. It should be noted that graph communities found by maximizing modularity should be judged carefully. While modularity is one natural measure of community structure in networks, there is no guarantee that it captures the particular structure relevant in a specific domain. For example, Fortunato and Barthélemy [20] have recently shown that modularity and more generally, each "quality function" (characterizing the quality of the entire partition in one number) have an intrinsic resolution scale, and can therefore fail to detect communities smaller than that scale. More fundamentally, Kleinberg [21] has shown that no single clustering method can ever satisfy four natural desiderata on all instances.

Recently, Brandes et al. [22] have shown that finding the clustering of maximum modularity for a given graph is NP-complete. This means that efficient algorithms to always find an optimal clustering, in time polynomial in the size of the graph for *all* graphs, are unlikely to exist. It is thus desirable to develop heuristics yielding clusterings as close to optimal as possible.

In this paper, we introduce the technique of solving and rounding fractional mathematical programs to the problem of community discovery, and propose two new algorithms for finding modularity-maximizing clusterings. The first algorithm is based on a linear programming (LP) relaxation of an integer programming (IP) formulation. The LP relaxation will put nodes "partially in the same cluster". We use a "rounding" procedure due to Charikar et al. [23] for the problem of *Correlation Clustering* [24]. The idea of the algorithm is to interpret "partial membership of the same cluster" as a distance metric, and group together nearby nodes.

The second algorithm is based on a vector programming (VP) relaxation of a quadratic program (QP). It recursively splits one partition into two smaller partitions while a better modularity can be obtained. It is similar in spirit to an approach recently proposed by Newman [6,18], which repeatedly divides clusters based on the first eigenvector of the modularity matrix. Newman's approach can be thought of as embedding nodes in the interval $[-1, 1]$, and then cutting the interval in the middle. The VP embeds nodes on the surface of a high-dimensional hypersphere, which is then randomly cut into two halves containing the nodes. The approach is thus very similar to the algorithm for Maximum Cut due to Goemans and Williamson [25].

A significant advantage of our algorithms over past approaches is that they provide an a posteriori guarantee on the error bound. The value obtained by the LP relaxation is an upper bound on the maximum achievable modularity. If the solution produced by our algorithm is within a factor of α of this upper bound, then we are guaranteed that it is also within a factor α of the *best possible* clustering. In principle, the bound provided by the LP could be loose; however, it was very accurate in all our test instances. Akin to the case of the LP relaxation, the value of the VP relaxation gives a bound on the best division of the graph into *two* communities.

We evaluate our algorithms on several standard test cases for graph community identification, comprising several real-world networks and a suite of networks generated according to a specified process for obtaining more or less pronounced community structure with given degree distributions. On every real-world test case where an upper bound on the optimal solution could be determined, the solution found using both our algorithms attains at least 99% of the theoretical upper bound; sometimes, it is optimal. On the structured random instances, the solutions were within 99% of the ground truth for pronounced clustering, within 97% for medium-pronounced clustering, and within 90% for very unpronounced ground truth clustering. In addition, both algorithms match or outperform past modularity maximization algorithms on most test cases, and match even exhaustive techniques such as Simulated Annealing, which requires significantly longer running time. Thus, our results suggest that these algorithms are excellent choices for finding graph communities.

The performance of our algorithms comes at a price of significantly slower running time and higher memory requirements than past heuristics. The bulk of both time and memory are consumed by the LP or VP solver; the rounding is comparatively simple. Mostly due to the high

memory requirements, the LP rounding algorithm can currently only be used on networks of up to a few hundred nodes. The VP rounding algorithm has lower running time and memory requirements than the LP method and scales to networks of up to a few thousand nodes on a personal desktop computer.

We believe that despite their lower efficiency, our algorithms provide three important contributions. First, they are the first algorithms with guaranteed polynomial running time to provide a posteriori performance guarantees. Second, they match or outperform past algorithms for medium-sized networks of practical interest. And third, the approach proposed in our paper introduces a new algorithmic paradigm to the physics community. Future work using these techniques would have the potential to produce more efficient algorithms with smaller resource requirements. Indeed, in the past, algorithms based on rounding LPs were often a first step towards achieving the same guarantees with purely combinatorial algorithms. Devising such algorithms is a direction of ongoing work.

2 Preliminaries

The network is given as an undirected graph $G = (V, E)$. The adjacency matrix of G is denoted by $A = (a_{u,v})$: thus, $a_{u,v} = a_{v,u} = 1$ if u and v share an edge, and $a_{u,v} = a_{v,u} = 0$ otherwise. The degree of a node v is denoted by d_v. A *clustering* $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a partition of V into disjoint sets C_i. We use $\gamma(v)$ to denote the (unique) index of the cluster that node v belongs to.

The *modularity* [14] of a clustering $\mathcal{C}$ is the total number of edges inside clusters, minus the expected number of such edges if the graph were a uniformly random multigraph subject to its degree sequence. In order to be able to compare the modularity for graphs of different sizes, it is convenient to normalize this difference by a factor of $1/2m$, so that the modularity is a number from the interval $[-1, 1]$.

If nodes u, v have degrees d_u, d_v, then any one of the m edges has probability $2\frac{d_u}{2m} \cdot \frac{d_v}{2m}$ of connecting u and v (the factor 2 arises because either endpoint of the edge could be u or v). By linearity of expectation, the expected number of edges between u and v is then $\frac{d_u d_v}{2m}$. Thus, the modularity of a clustering $\mathcal{C}$ is

$$Q(\mathcal{C}) := \frac{1}{2m} \sum_{u,v} \left(a_{u,v} - \frac{d_u d_v}{2m} \right) \cdot \delta(\gamma(u), \gamma(v)), \quad (1)$$

where δ denotes the Kronecker Delta, which is 1 if its arguments are identical, and 0 otherwise. Newman [6] terms the matrix M with entries $m_{u,v} := a_{u,v} - \frac{d_u d_v}{2m}$ the *modularity matrix* of G. For a more detailed discussion of the probabilistic interpretation of modularity and generalizations of the measure, see the recent paper by Gaertler et al. [26].

3 Algorithms

3.1 Linear programming based algorithm

3.1.1 The linear program

Based in equation (1), we can phrase the modularity maximization problem as an integer linear program (IP). (For an introduction to Linear Programming, we refer the reader to [27,28]; for the technique of LP rounding, see [29].) The linear program has one variable $x_{u,v}$ for each pair (u, v) of vertices. We interpret $x_{u,v} = 0$ to mean that u and v belong to the same cluster, and $x_{u,v} = 1$ that u and v are in different clusters. Then, the objective function to be maximized can be written as $\sum_{u,v} m_{u,v}(1 - x_{u,v})$. This is a linear function, because the $m_{u,v}$ are constants. We need to ensure that the $x_{u,v}$ are consistent with each other: if u and v are in the same cluster, and v and w are in the same cluster, then so are u and w. This constraint can be written as a linear inequality $x_{u,w} \leq x_{u,v} + x_{v,w}$. It is not difficult to see that the $x_{u,v}$ are consistent (i.e., define a clustering) if and only if this inequality holds for all triples (u, v, w). Thus, we obtain the following integer linear program (IP):

$$\begin{aligned} &\text{Maximize} \quad \frac{1}{2m} \cdot \textstyle\sum_{u,v} m_{u,v} \cdot (1 - x_{u,v}) \\ &\text{subject to} \quad x_{u,w} \leq x_{u,v} + x_{v,w} \text{ for all } u, v, w \\ &\qquad\qquad x_{u,v} \in \{0, 1\} \quad \text{for all } u, v. \end{aligned} \quad (2)$$

Solving IPs is also NP-hard, and thus unlikely to be possible in polynomial time. However, by replacing the last constraint – that each $x_{u,v}$ be an integer from $\{0, 1\}$ – with the constraint that each $x_{u,v}$ be a real number between 0 and 1, we obtain a linear program (LP). LPs can be solved in polynomial time [28,30], and even quite efficiently in practice. (For our experiments, we use the widely used commercial package CPLEX.) The downside is that the solution, being fractional, does not correspond to a clustering. As a result, we have to apply a post-processing step, called "rounding" of the LP.

3.1.2 The LP rounding algorithm

Our LP rounding algorithm is essentially identical to one proposed by Charikar et al. [23] for the *correlation clustering* problem [24]. In correlation clustering, one is given an undirected graph $G = (V, E)$ with each edge labeled either '+' (modeling similarity between endpoints) or '−' (modeling dissimilarity). The goal is to partition the graph into clusters such that few vertex pairs are classified incorrectly. Formally, in the MinDisagree version of the problem, the goal is to minimize the number of '−' edges inside clusters plus the number of '+' edges between clusters. In the MaxAgree version, which is not as relevant to our approach, the goal is to maximize the number of '+' edges inside clusters plus the number of '−' edges between clusters. Using the same 0–1 variables $x_{u,v}$ as we

did above, Charikar et al. [23] formulate MinDisagree as follows:

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{(u,v)\in E_+} x_{u,v} + \sum_{(u,v)\in E_-} (1 - x_{u,v}) \\ &\text{subject to} && x_{u,w} \le x_{u,v} + x_{v,w} \quad \text{for all } u, v, w \\ & && x_{u,v} \in \{0, 1\} \quad\quad \text{for all } u, v, \end{aligned}$$

where E_+ and E_- denote the sets of edges labeled '+' and '−', respectively. The objective can be rewritten as $|E_+| - \sum_{(u,v)\in E} \mu_{u,v}(1 - x_{u,v})$, where $\mu_{u,v}$ is 1 for '+' edges and –1 for '−' edges. The objective is minimized when $\sum_{(u,v)\in E} \mu_{u,v}(1 - x_{u,v})$ is maximized; thus, except for the shift by the constant $|E_+|$, MinDisagree takes on the same form as modularity maximization with $m_{u,v} = \mu_{u,v}$.

The rounding algorithm proposed by Charikar et al. [23] comes with an a priori error guarantee that the objective produced is never more than 4 times the optimum. Algorithms with such guarantees are called *approximation algorithms* [29], and it would be desirable to design such algorithms for modularity maximization as well. Unfortunately, the shift by a constant prevents the approximation guarantees from [23] from carrying over to the modularity maximization problem. However, the analogy suggests that algorithms for rounding the solution to the MinDisagree LP may perform well in practice for modularity maximization.

Our rounding algorithm, based on the one by Charikar et al., first solves the linear program (2) without the integrality constraints. This leads to a *fractional* assignment $x_{u,v}$ for every pair of vertices. The LP constraints, applied to fractional values $x_{u,v}$, exactly correspond to the triangle inequality. Hence, the $x_{u,v}$ form a metric, and we can interpret them as "distances" between the vertices. We use these distances to repeatedly find clusters of "nearby" nodes, which are then removed. The full algorithm is as follows:

Algorithm 1 Modularity Maximization Rounding

1: Let $S = V$.
2: **while** S is not empty **do**
3: Select a vertex u from S.
4: Let T_u be the set of vertices whose distance from u is at most $\frac{1}{2}$.
5: **if** the average distance of the vertices in $T_u \setminus \{u\}$ from u is less than $\frac{1}{4}$ **then**
6: Make $C = T_u$ a cluster.
7: **else**
8: Make $C = \{u\}$ a singleton cluster.
9: **end if**
10: Let $S = S \setminus C$.
11: **end while**

Step 3 of the rounding algorithm is underspecified: it does not say *which* of the remaining vertices u to choose as a center next. We found that selecting a random center in each iteration, and keeping the best among 1000 independent executions of the entire rounding algorithm, significantly outperformed two natural alternatives, namely selecting the largest or smallest cluster. In particular, selecting the largest cluster is a significantly inferior heuristic.

As a post-processing step to the LP rounding, we run a *local-search* algorithm proposed by Newman [6] to refine the results further. The post-processing step is briefly described in Section 3.3.

An important benefit of the LP rounding method is that it provides an upper bound on the best solution. For the best clustering is the optimum solution to the integer LP (2); removing the integrality constraint can only increase the set of allowable solutions to the LP, improving the objective value that can be obtained. The upper bound enables us to lower-bound the performance of clustering algorithms.

The other useful feature of our algorithm is its inherent capability to find different clusterings with similar modularity. The randomization naturally leads to different solutions, of which several with highest modularity values can be retained, to provide a more complete picture of possible cluster boundaries.

3.2 Vector programming based algorithm

In this section, we present a second algorithm which is more efficient in practice, at the cost of slightly reduced performance. It produces a "hierarchical clustering", in the sense that the clustering is obtained by repeatedly finding a near-optimal division of a larger cluster. For two reasons, this clustering is not truly hierarchical: first, we do not seek to optimize a global function of the entire hierarchy, but rather optimize each split locally. Second, we again apply a local search based post-processing step to improve the solution, thus rearranging the clusters. Despite multiple recently proposed hierarchical clustering algorithms (e.g., [6,8,31]), there is far from general agreement on what objective functions would capture a "good" hierarchical clustering. Indeed, different objective functions can lead to significantly different clusterings. While our clustering is not truly hierarchical, the order and position of the splits that it produces still reveal much high-level information about the network and its clusters.

As discussed above, our approach is to aim for the best division *at each level individually*, requiring a partition into two clusters at each level. Clusters are recursively subdivided as long as an improvement is possible. Thus, a solution hinges on being able to find a good partition of a given graph into *two* communities. The LP rounding algorithm presented in the previous section is not applicable to this problem, as it does not permit specifying the number of communities. Instead, we will use a Vector Programming (VP) relaxation of a Quadratic Program (QP) to find a good partition of a graph into two communities.

3.2.1 The quadratic program

Our approach is motivated by the same observation that led Newman [6] to an eigenvector-based partitioning approach. For every vertex v, we have a variable y_v which

is 1 or –1 depending on whether the vertex is in one or the other partition. Since each pair u, v adds $m_{u,v}$ to the objective if u and v are in the same partition (and zero otherwise), the objective function can be written as $\frac{1}{4m}\sum_{u,v} m_{u,v}(1 + y_u y_v)$. Newman [6] rewrites this term further as $\frac{1}{4m}\mathbf{y}^T M \mathbf{y}$ (where $\mathbf{y}$ is the vector of all y_v values), and observes that if the entries y_v were not restricted to be ± 1, then the optimal $\mathbf{y}$ would be the principal eigenvector of M. His approach, in line with standard spectral partitioning approaches (e.g., [32]), is then to compute the principal eigenvector $\mathbf{y}$, and partition the nodes into positive y_v and negative y_v. Thus, in a sense, Newman's approach can be considered as embedding the nodes optimally on the line, and then rounding the fractional solution into nodes with positive and negative coordinates.

Our solution also first embeds the nodes into a metric space, and then rounds the locations to obtain two communities. However, it is motivated by considering the objective function as a strict quadratic program (see, e.g., [29]). We can write the problem of partitioning the graph into two communities of maximum modularity as

$$\begin{aligned} &\text{Maximize } \frac{1}{4m}\sum_{u,v} m_{u,v}(1 + y_u y_v) \\ &\text{subject to } y_v^2 = 1 \text{ for all } v. \end{aligned} \tag{3}$$

Notice that the constraint $y_v^2 = 1$ ensures that each y_v is ± 1 in a solution to (3).

Quadratic Programming, too, is NP-complete. Hence, we use the standard technique of relaxing the QP (3) to a corresponding Vector Program (VP), which in turn can be solved in polynomial time using semi-definite programming (SDP). To turn a strict quadratic program into a vector program, one replaces each variable y_v with a (n-dimensional) vector-valued variable $\mathbf{y}_v$, and each product $y_u y_v$ with the inner product $\mathbf{y}_u \cdot \mathbf{y}_v$. We use the standard process [29] for transforming the VP formulation to the SDP formulation and for obtaining back the solution to the VP from the solution to SDP. For solving the SDP problems in our experiments, we use a standard off-the-shelf solver CSDP [33].

The result of solving the VP will be vectors $\mathbf{y}_v$ for all vertices v, which can be interpreted as an embedding of the nodes on the surface of the hypersphere in n dimensions. (The constraint $\mathbf{y}_v \cdot \mathbf{y}_v = 1$ for all v ensures that all nodes are embedded at distance 1 from the origin.) Thus, the inner product of two node positions $\mathbf{y}_u, \mathbf{y}_v$ is equal to the cosine of the angle between them. As a result, the optimal VP solution will "tend to" have node pairs with negative $m_{u,v}$ far apart (large angles), and node pairs with positive $m_{u,v}$ close (small angles).

3.2.2 Rounding the quadratic program

To obtain a partition from the node locations $\mathbf{y}_v$, we use a rounding procedure proposed by Goemans and Williamson [25] for the Max-Cut problem. In the Max-Cut problem, an undirected graph is to be partitioned into two disjoint node sets so as to maximize the number of edges crossing between them. This objective can be written as a quadratic program as follows (notice the similarity to the Modularity Maximization QP):

$$\begin{aligned} &\text{Maximize } \frac{1}{2}\sum_{(u,v)\in E}(1 - y_u y_v) \\ &\text{subject to } y_v^2 = 1 \text{ for all } v. \end{aligned}$$

The rounding procedure of Goemans and Williamson [25], which we adopt here, chooses a random $(n-1)$-dimensional hyperplane passing through the origin, and uses the hyperplane to cut the hypersphere into two halves. The two partitions are formed by picking the vertices lying on each side of the hypersphere. The cutting hyperplane is represented by its normal vector $\mathbf{s}$, which is an n-dimensional vector, each of whose components is an independent $\mathcal{N}(0,1)$ Gaussian. (It is well known and easy to verify that this makes the direction of the normal uniformly random.) To cut the hypersphere, we simply define $S := \{v \mid \mathbf{y}_v \cdot \mathbf{s} \geq 0\}$ and $\overline{S} := \{v \mid \mathbf{y}_v \cdot \mathbf{s} < 0\}$. Once the VP has been solved (which is the expensive part), one can easily choose multiple random hyperplanes, and retain the best resulting partition. In our experiments, we chose the best of 5000 hyperplanes.

A different approach to rounding VP solutions of the form (3) was recently proposed by Charikar and Wirth [34], again in the context of Correlation Clustering. Their method first projects the hypersphere on a random line, scales down large coordinates, and then rounds randomly. Their method gives an a priori error guarantee of $\Omega(1/\log n)$ under the assumption that all diagonal entries of the matrix M are zero. In fact, if the matrix is also positive semi-definite, then a result of Nesterov [35] shows that the approximation guarantee can be improved to $2/\pi$. Unfortunately, the modularity matrix M is neither positive semi-definite nor does it have an all-zero diagonal; hence, neither of these approximation results is applicable to the problem of finding the modularity-maximizing partition into two communities.

We also implemented the rounding procedure of [34], and tested it on the same example networks as the other algorithms. We found that its performance is always inferior to the hyperplane based algorithm, sometimes significantly so. Since the algorithm is not more efficient, either, we omit the results from our comparison in Section 4.

3.2.3 The hierarchical clustering algorithm

Note that the effect of partitioning a community C further into two sub-communities C', C'' is independent of the structure of the remaining communities, because any edge inside one of the other communities remains inside, and the *expected number* of edges inside other communities also stays the same. Thus, in splitting C into C' and C'', the modularity Q increases by

$$\Delta Q(C) = \frac{1}{m}\left(\frac{(\sum_{v\in C'} d_v)(\sum_{u\in C''} d_u)}{2m} - |e(C', C'')|\right),$$

where $e(C', C'')$ denotes the set of edges between C' and C''.

The target communities C', C'' are calculated using the above VP rounding, and the algorithm will terminate when none of the $\Delta Q(C)$ are positive. The full algorithm is given below.

The use of a Max-Heap is not strictly necessary; a set of active communities would have been sufficient. However, the choice of a Max-Heap has the added advantage that by slightly tweaking the termination condition (requiring an increase greater than some ϵ), one can force the communities to be larger, and the algorithm to terminate faster.

It is important that in each iteration of the algorithm, the degrees d_v for each vertex v and the total number of edges m be calculated by taking into account all the edges in the entire graph and not just the edges belonging to the sub-graph being partitioned.

Algorithm 2 Hierarchical Clustering

1: Let M be an empty Max-Heap.
2: Let C be a cluster containing all the vertices.
3: Use VP rounding to calculate (approximately) the maximum increase in modularity possible, $\Delta Q(C)$, achievable by dividing C into two partitions.
4: Add $(C, \Delta Q(C))$ to M.
5: **while** the head element in M has $\Delta Q(C) > 0$ **do**
6: Let C be the head of M.
7: Use VP rounding to split C into two partitions C', C'', and calculate $\Delta Q(C'), \Delta Q(C'')$.
8: Remove C from M.
9: Add $(C', \Delta Q(C')), (C'', \Delta Q(C''))$ to M.
10: **end while**
11: Output as the final partitioning all the partitions remaining in the heap M, as well as the hierarchy produced.

As a post-processing step, we run the *local-search* algorithm proposed by Newman [6]. The post-processing brings the VP results nearly to par with those obtained by the LP method.

3.3 Local search algorithm

We use the local-search algorithm proposed by Newman [6] for refining the results obtained by our LP and VP methods. This method improved the modularity of the partitions produced by the LP method by less than 1% and in the case of the QP method, it improved the modularity by less than 5%. The local search method is based on the Kernighan-Lin algorithm for graph bisection [36]. Starting from some initial network clustering, the modularity is iteratively improved as follows: select the vertex which, when moved to another group, results in the maximum increase in modularity (or minimum decrease, if no increase is possible). In one complete iteration, each vertex changes its group exactly once; at the end of the iteration, the intermediate clustering with the highest modularity value is selected as the new clustering. This process is continued as long as there is an increase in the overall modularity. For details of the implementation, we refer the reader to [6].

4 Examples

In this section, we present results for both of our algorithms on several real-world and synthetic networks. We focus on well-studied and structured random networks since our goal in this paper is to compare the quality of optimization achieved by our methods to approaches in past work, rather than discovering novel structure. We restrict our attention here to networks with at most a few thousand nodes, as this is currently the limit for our algorithms. The algorithm implementations are available online at `http://www-scf.usc.edu/~gaurava`. All our experiments were carried out on a Linux-based Intel PC with two 3.2 GHz processors and 2 GB of RAM.

We evaluate our results in two ways: manually and by comparing against past work. For several smaller networks, we show below the clusterings obtained by the LP rounding algorithm. In all of the cases, the clusterings can be seen to be closely correlated with some known "semantic" information about the network. We then evaluate the algorithms on structured random networks generated according to a process suggested by Lancichinetti et al. [37], and finally present an extensive comparison of the results of our algorithms with those of past heuristics and Simulated Annealing.

4.1 Zachary's Karate club

The "*Zachary's Karate Club*" [38] network represents the friendships between 34 members of a karate club in the US over a period of 2 years. It has come to be a standard test network for clustering algorithms, partly due to the fact that during the observation period, the club broke up into two separate clubs over a conflict, and the resulting two new clubs can be considered a "ground truth" clustering. Both of our algorithms find a community structure identical to the one detected by Medus et al. [39]. It has a modularity of 0.4197. Our algorithm also proves this value to be best possible, because the LP returned a $\{0, 1\}$-solution, i.e., no rounding was necessary. The community structure found for the Karate Club network is shown in Figure 1.

For finding the primary two-community division in this network, we ran a single iteration of the VP algorithm and found a partition identical to that found by Medus et al. [39]. This partition corresponds almost exactly to the actual factions in the club, with the exception of node 10. The bipartition found by the VP method has a modularity of 0.3718, whereas the partition corresponding to the actual factions in the club has a lower modularity of 0.3715. This explains the "misclassification" of node 10, and also emphasizes that no clustering objective can be guaranteed to always recover the "semantically correct" community structure in a real network. The latter should

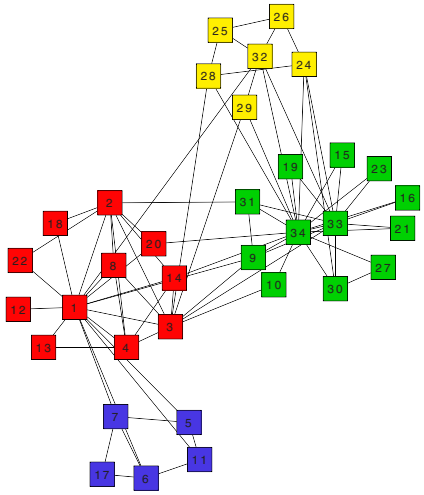

Fig. 1. The optimal community structure with modularity 0.4197 for Zachary's Karate Club network. Each community is shaded with a different color.

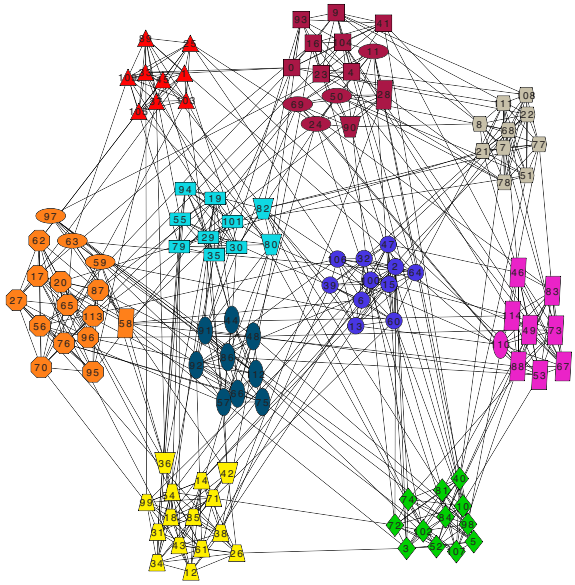

Fig. 2. The partitioning of the College Football network found by the LP rounding algorithm. Each detected community is shaded with a different color. The actual conferences are depicted using different shapes.

be taken as a cautioning against accepting modularity-maximizing clusterings as ground truth.

4.2 College football

This data set representing the schedule of division I football games for the 2000 season was compiled by Girvan and Newman [8]. Vertices in the graph represent teams, and edges represent regular season games between the two teams they connect. The teams are divided into *conferences* with 8–12 teams each. Usually, more games are played within conferences than across conferences, and it is an interesting question whether the ground truth of conferences can be reconstructed by observing the games played. Both our algorithms find the same clustering with modularity 0.6046, shown in Figure 2. The algorithms accurately recover most of the conferences as well as the independent teams (which do not belong to any conference).

Our algorithms also found a slightly suboptimal clustering of modularity 0.6044, combining two prominent conferences, *Mountain West* and *Pacific* 10 (brown squares and gray hexagons in the top right corner) into one community. The reason is that many games were played between teams of the two conferences. This shows that community detection is inherently unstable: solutions with only slightly different modularity (differing only by 0.0002) can differ significantly. Such slight differences could easily elude heuristic algorithms. More importantly, this instability shows again that communities maximizing modularity should be evaluated carefully for semantic relevance. With respect to such instabilities in community structure, Gfeller et al. [40] give a more detailed analysis and provide methods for detecting them. However, their methods are applicable only to non-randomized clustering algorithms.

This example illustrates an advantage of our randomized rounding algorithms, which produce multiple different solutions. These solutions together often reveal more information about community boundaries. They can also be manually inspected if desired, and a researcher with domain knowledge can pick the one representing the true underlying structure most accurately.

4.3 Books on american politics

As a final example, Figure 3 shows the community structure detected in the *American Political Books* network compiled by Krebs. The vertices represent books on American politics bought from `amazon.com`, and edges connect pairs of books frequently co-purchased. The books in this network were classified by Newman [18] into categories liberal or conservative, except for a small number of books with no clear ideological leaning. Figure 3 shows that our algorithm accurately detects a strong community structure, which matches fairly well the underlying semantic division based on political slant.

The community structure produced by our LP algorithm has a modularity of 0.5272 and agrees mostly with the manual labeling. It is very similar to the one produced by Newman [6], except for an extra cluster of three nodes produced by our method, as well as slightly fewer "misclassified" nodes in the two main clusters. The three books in the additional cluster were biographical in nature, and were always bought together. The additional cluster is not found by the VP method, which instead merges the three biographical books with the blue cluster, and obtains a modularity of 0.5269.

For finding the primary division in this network, we ran a single iteration of the VP algorithm. The partition has a modularity value of 0.4569. It produces a partition with all the *liberal books* and three of the *conservative books*

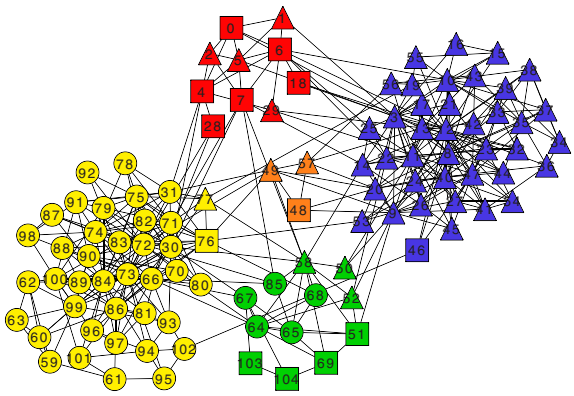

Fig. 3. The partitioning of the American Political Books network found by the LP rounding algorithm. Each detected community is shaded with a different color, while actual political slants are depicted using different shapes. The circles are *liberal books*, the triangles are *conservative books*, and the squares are *centrist*.

assigned to one cluster and the remaining *conservative books* assigned to the other cluster. The *centrist books* were divided roughly evenly among the two clusters.

We also computed the modularity values for various "ground truth" partitionings. If the books are divided into three communities corresponding to liberal, conservative, and centrist (according to a manual labeling), the modularity is significantly inferior to our best clustering, namely 0.4149. If the centrist books are completely grouped with either the liberal or conservative books, the modularity deteriorates further to 0.3951 resp. 0.4088, which is noticeably worse than the modularity of 0.4569 achieved by the bipartition of our algorithm. This corroborates an observation already made in discussing the Zachary Karate Club: the semantic ground truth partitioning will not necessarily achieve the highest modularity as a network partition, and hence, the two should not be treated as identical.

4.4 Structured random graphs

In addition to the real-world graphs described above, we also tested our algorithms on several graphs drawn from a distribution of representative random graphs, as suggested recently by Lancichinetti et al. [37]. This family extends the basic idea of planted partitionings as suggested by McSherry [41] and Girvan and Newman [8], to generate a variable number of communities, with parametrized degree distributions (parameter γ), community distributions (parameter β), average node degree (k), and mixing parameters (μ). The mixing parameter μ controls how pronounced communities are, i.e., how much more likely edges are within communities than across communities. For a detailed discussion of the meaning of these parameters, see [37].

We report the results for six representative graphs in Table 1 below. For each instance, we report the parameter settings (n is the total number of nodes). We include the number of communities in the ground truth (GT-c) as well as the number output by the VP algorithm (VP-c). Similarly, we report the modularity of the ground truth clustering (GT-m) and of the clustering found by the VP rounding (VP-m). Finally, we also include the percentage of nodes that were clustered according to the ground truth by the VP algorithm.

Table 1. Ground truth clustering and clustering obtained by VP rounding for several structured random graphs.

n	γ	β	k	μ	GT-c	VP-c	GT-m	VP-m	% agree
500	2	1	20	0.2	14	14	0.710	0.710	100
1000	2	1	20	0.2	31	30	0.746	0.745	97.5
500	3	2	20	0.4	15	15	0.514	0.512	99.6
1000	3	2	20	0.4	30	30	0.556	0.541	97.6
500	2	1	20	0.6	16	15	0.326	0.295	79.2
1000	2	1	20	0.6	31	19	0.359	0.321	56.9

From the table, it is evident that the VP algorithm identifies communities quite accurately when the community structure is pronounced (small or medium range μ parameters), but its performance deteriorates when there are many edges between different communities. In that case, it tends to merge multiple smaller communities into larger ones, which is in accordance with the predictions of Fortunato and Barthélemy concerning resolution limits of community identification [20]. Notice also that for the range $\mu = 0.6$, even the modularity of the ground truth clustering is comparable to that of an Erdős-Rényi random graph [42].

4.5 Other examples and running times

We tested our methods on several other networks and were able to identify community structures with very high modularity values. The test networks included a collaboration network of *jazz musicians* (JAZZ) [43], the social network of a community of 62 *bottlenose dolphins* (DOLPH) living in Doubtful Sound, New Zealand [44], an interaction network of the characters from Victor Hugo's novel *Les Misérables* (MIS) [45], a *collaboration network* (COLL) of scientists who conduct research on networks [46], a *metabolic* network for the nematode *C.elegans* (META) [47] and a network of *email contacts* between students and faculty (EMAIL) [48].

We compare our algorithms against past published partitioning heuristics, specifically, the edge-betweenness based algorithm of Girvan and Newman [8] (denoted by GN), the extremal optimization algorithm of Duch and Arenas [10] (DA), and the eigenvector based algorithm of Newman [6,18]. We exclude the bottom-up heuristic of Clauset, Moore, and Newman [16], since it is designed not so much to yield close-to-optimal clusterings as to give reasonable clusterings for extremely large networks (several orders of magnitude beyond what our algorithms can deal with); the performance of this heuristic is significantly inferior to the other methods.

In addition to these heuristics, we also compare our algorithms to a Simulated Annealing implementation (SA)

Table 2. The modularity obtained by many of the previously published methods and by the methods introduced in this paper, along with the upper bound.

NETWORK	size n	GN	DA	EIG	VP	LP	UB
KARATE	34	0.401	0.419	0.419	0.420	0.420	0.420
DOLPH	62	0.520	–	–	0.526	0.529	0.531
MIS	76	0.540	–	–	0.560	0.560	0.561
BOOKS	105	–	–	0.526	0.527	0.527	0.528
BALL	115	0.601	–	–	0.605	0.605	0.606
JAZZ	198	0.405	0.445	0.442	0.445	0.445	0.446
COLL	235	0.720	–	–	0.803	0.803	0.805
META	453	0.403	0.434	0.435	0.450	–	–
EMAIL	1133	0.532	0.574	0.572	0.579	–	–

Table 3. The modularity and running times (in minutes and seconds) of our algorithms as well as Simulated Annealing with different cooling schedules.

Network	SA (0.999)	SA (0.95)	VP	LP
KARATE	0.420 [0:12]	0.420 [0:02]	0.420 [0:06]	[0:02]
DOLPH	0.528 [2:55]	0.527 [0:05]	0.526 [0:09]	[0:04]
MIS	0.560 [4:22]	0.556 [0:10]	0.560 [0:11]	[0:04]
BOOKS	0.527 [13:02]	0.527 [0:26]	0.527 [0:12]	[0:28]
BALL	0.605 [4:10]	0.604 [0:06]	0.605 [0:23]	[0:18]
JAZZ	0.445 [58:05]	0.445 [2:50]	0.445 [0:24]	[29:22]
COLL	0.799 [25]	0.799 [0:32]	0.803 [1:45]	[32:21]
META	0.450 [146]	0.445 [9:02]	0.450 [1:30]	–
EMAIL	0.579 [1143]	0.575 [40:12]	0.579 [15:08]	–

by Guimerà and Amaral [49], representing a more exhaustive and slower search. We use three different settings for the cooling schedule, in the range [0.95,0.999] recommended by Guimerà et al. These cover the range of matching the quality of our clustering to matching the running time of our algorithms.

We summarize the results in Tables 2 and 3. Table 2 contains the results of our algorithm and of the past heuristics, as well as the upper bound on modularity obtained by the linear program. Table 3 compares our results and running times to those of Simulated Annealing with different cooling schedules. Some LP heuristic and upper bound entries for larger data sets are missing, because the LP solver could not solve such large instances.

Noticeably, both the LP and VP rounding algorithms outperformed all past heuristics in terms of the value of modularity obtained. The upper bound provided by the LP shows that the modularity obtained by our algorithm is close to optimal for all examples. Notice that it is not clear whether the upper bound can in fact be attained by any clustering; it seems quite plausible that the clustering produced by our algorithms is in fact optimal.

Since the running time of the LP and VP rounding algorithms is significantly larger than for past heuristics, we also compare them with Simulated Annealing [49], a slower and more exhaustive algorithm. For this comparison, we report both the modularity values obtained and the running time of the different algorithms. For Simulated Annealing, we chose three cooling schedules, 0.999, 0.99 and 0.95. As mentioned above, all the running times were measured on a Linux-based Intel PC with two 3.2 GHz processors and 2 GB of RAM. For readability, we omit from the table below the modularity obtained by the LP algorithm, which is given in Table 2 and identical to the one for the VP algorithm, except for the DOLPH network. We also omit the results for the cooling schedule of 0.99. Both the modularity obtained and the running time were between the ones for 0.999 and 0.95.

Notice that the results obtained by our algorithm are only inferior for one data set to Simulated Annealing with the slowest cooling schedule. For all other data sets and schedules, our algorithms match or outperform Simulated Annealing, even while taking comparable or less time than the faster cooling schedules.

5 Conclusion

We have shown that the technique of rounding solutions to fractional mathematical programs yields high-quality modularity maximizing communities, while also providing a useful upper bound on the best possible modularity. The drawback of our algorithms is their resource requirement. Due to $\Theta(n^3)$ constraints in the LP, and $\Theta(n^2)$ variables in the VP, the algorithms currently do not scale beyond about 300 resp. 4000 nodes. Thus, a central goal for future work would be to improve the running time without sacrificing solution quality. An ideal outcome would be a purely combinatorial algorithm avoiding the explicit solution to the mathematical programs, but yielding the same performance.

Secondly, while our algorithms perform very well on all networks we considered, they do not come with *a priori guarantees* on their performance. Heuristics with such performance guarantees are called *approximation algorithms* [29], and are desirable because they give the user a hard guarantee on the solution quality, even for pathological networks. Since the algorithms of Charikar et al. and Goemans and Williamson on which our approaches are based do have provable approximation guarantees, one would hope that similar guarantees could be attained for modularity maximization. However, this does not hold for the particular algorithms we use, due to the shift of the objective function by a constant. Obtaining approximation algorithms for modularity maximization thus remains a challenging direction for future work.

We would like to thank Aaron Clauset, Cris Moore, Mark Newman and Ashish Vaswani for useful discussions and advice, Fernando Ordóñez for providing computational resources, and Roger Guimerà for sharing his implementation of Simulated Annealing. We also thank several anonymous reviewers for helpful feedback on previous versions of the paper. David Kempe has been supported in part by NSF CAREER Award 0545855.

References

1. M. Newman, A. Barabási, D. Watts, *The Structure and Dynamics of Networks* (Princeton University Press, 2006)
2. J. Scott, *Social Network Analysis: A Handbook*, 2nd edn. (Sage Publications, 2000)
3. S. Wasserman, K. Faust, *Social Network Analysis* (Cambridge University Press, 1994)
4. J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, The web as a graph: Measurements, models and methods, in *International Conference on Combinatorics and Computing*, 1999
5. R. Guimerà, L. Amaral, Nature **433**, 895 (2005)
6. M. Newman, Phys. Rev. E **74**, 036104 (2006)
7. G. Flake, S. Lawrence, C.L. Giles, F. Coetzee, IEEE Computer **35**, 66 (2002)
8. M. Girvan, M. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002)
9. M. Newman, Eur. Phys. J. B **38**, 321 (2004)
10. J. Duch, A. Arenas, Phys. Rev. E **72**, 027104 (2005)
11. G. Flake, R. Tarjan, K. Tsioutsiouliklis, *Graph clustering techniques based on minimum cut trees*, Technical Report 2002-06 (NEC, Princeton, 2002)
12. A. Hayrapetyan, D. Kempe, M. Pál, Z. Svitkina, Unbalanced graph cuts, in *Proc. 13th European Symp. on Algorithms*, 2005, pp. 191–202
13. M. Charikar, Greedy approximation algorithms for finding dense components in graphs, in *Proc. 3rd Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2000
14. M. Newman, M. Girvan, Phys. Rev. E **69**, 026113 (2004)
15. M. Newman, Phys. Rev. E **69**, 066133 (2004)
16. A. Clauset, M. Newman, C. Moore, Phys. Rev. E **70**, 066111 (2004)
17. A. Clauset, Phys. Rev. E **72**, 026132 (2005)
18. M. Newman, Proc. Natl. Acad. Sci. USA **103**, 8577 (2006)
19. L. Danon, J. Duch, A. Diaz-Guilera, A. Arenas, J. Stat. Mech. P09008 (2005)
20. S. Fortunato, M. Barthélemy, Proc. Natl. Acad. Sci. USA **104**, 36 (2007)
21. J. Kleinberg, *An impossibility theorem for clustering*, in *Proc. Advances in Neural Information Processing Systems (NIPS)* (2002)
22. U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, IEEE Trans. Know, Data Eng. **20**, 172 (2008)
23. M. Charikar, V. Guruswami, A. Wirth, J. Comput. System Sci. 360 (2005)
24. N. Bansal, A. Blum, S. Chawla, Machine Learning **56**, 89 (2004)
25. M. Goemans, D. Williamson, J. ACM **42**, 1115 (1995)
26. M. Gaertler, R. Görke, D. Wagner, Significance-driven graph clustering, in *Proc. 3rd Intl. Conf. on Algorithmic Aspects in Information and Management*, 2007, pp. 11–26
27. V. Chvátal, *Linear Programming* (Freeman, 1983)
28. H. Karloff, *Linear Programming* (Birkhäuser, 1991)
29. V. Vazirani, *Approximation Algorithms* (Springer, 2001)
30. N. Karmarkar, Combinatorica **4**, 373 (1984)
31. M. Sales-Pardo, R. Guimera, A. Moreira, L. Amaral, Proc. Natl. Acad. Sci. USA **104**, 15224 (2007)
32. M. Fiedler, Czech. Math. J. **25**, 619 (1975)
33. B. Borchers, Optim. Meth. Softw. **11**, 613 (1999)
34. M. Charikar, A. Wirth, Maximizing quadratic programs: Extending grothendieck's inequality, in *Proc. 45th IEEE Symp. on Foundations of Computer Science*, 2004, pp. 54–60
35. Y. Nesterov, Optim. Meth. Softw. **9**, 141 (1998)
36. B. Kernighan, S. Lin, Bell Systems Tech. J. **49**, 291 (1970)
37. A. Lancichinetti, S. Fortunato, F. Radicchi, *New benchmark in community detection*, 2008, eprint `arXiv:0805.4770`
38. W. Zachary, J. Anthropol. Res. **33**, (1977)
39. A. Medus, G. Acuña, C. Dorso, Phys. A Stat. Mech. Appl. **358**, 593 (2005)
40. D. Gfeller, J.-C. Chappelier, P. De Los Rios, Phys. Rev. E **72**, (2005)
41. F. McSherry, Spectral partitioning of random graphs, in *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001, pp. 529–537
42. R. Guimerà, M. Sales-Pardo, L. Amaral, Phys. Rev. E, **70**, 025101 (2004)
43. P. Gleiser, L. Danon, Advances in Complex Systems **6**, 565 (2003)
44. D. Lusseau, Proc. of the Royal Society of London B **270**, 186 (2003)
45. D. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Algorithms* (ACM Press 1993)
46. M. Newman, Phys. Rev. E **64**, (2001)
47. H. Jeong, B. Tomber, R. Albert, Z. Oltvai, A.-L. Barabási, Nature **407**, 651 (2000)
48. R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, A. Arenas, Phys. Rev. E **68**, 065103 (2003)
49. R. Guimerà, L. Amaral, J. Stat. Mech. P02001 (2005)