Internet Mathematics, 11:181–200, 2015
Copyright © Taylor & Francis Group, LLC
ISSN: 1542-7951 print/1944-9488 online
DOI: 10.1080/15427951.2014.950875

TOWARD OPTIMAL COMMUNITY DETECTION: FROM TREES TO GENERAL WEIGHTED NETWORKS

Thang N. Dinh[1] and My T. Thai[2]
[1]*Computer Science Department, Virginia Commonwealth University, Richmond, Virginia, USA*
[2]*Computer & Information Science & Engineering, University of Florida, Gainesville, Florida, USA*

Abstract *Many networks, including the Internet, social networks, and biological relations, are found to be naturally divided into communities of densely connected nodes, known as community structure. Since Newman's suggestion of using* modularity *as a measure to qualify the goodness of community structures, many efficient methods to maximize modularity have been proposed but without* optimality guarantees. *In this work we study exact and theoretically near-optimal algorithms for maximizing modularity. In the first part, we investigate the complexity and approximability of the problem on tree graphs. Somewhat surprisingly, the problem is still NP-complete on trees. We then provide a polynomial time algorithm for uniform-weighted trees and a pseudopolynomial time algorithm and a PTAS for trees with arbitrary weights. In the second part, we present a family of compact linear programming formulations for the problem in general graphs. These formulations exploit the graph connectivity structure and reduce substantially the number of constraints, thus, they vastly improve the running time for solving linear programming and integer programming. As a result, networks of thousands of vertices can be solved in minutes, whereas the current largest instance solved with mathematical programming has fewer than 250 vertices.*

1. INTRODUCTION

Many complex systems of interest such as the Internet, social, networking, and biological relations can be represented as networks consisting of sets of *nodes* that are connected by *edges* between them. Research in a number of academic fields has uncovered unexpected structural properties of complex networks, including small-world phenomena [29]; power-law degree distribution [5]; and the existence of community structure [24] where in nodes are naturally clustered into tightly connected modules, also known as *communities*, with only sparser connections between them. Finding this community structure is a fundamental but challenging problem in the study of network systems and is not yet satisfactorily solved, despite the huge effort of a large interdisciplinary community of scientists over the past few years [14].

The ability to detect such communities can be of significant practical importance, providing insight into how network function and topology affect each other. For instance,

Address correspondence to Thang N. Dinh, Computer Science Department, Virginia Commonwealth University, Richmond, VA 23220, USA. E-mail: tndinh@vcu.edu

communities within the World Wide Web may correspond to sets of webpages on related topics; communities within mobile networks may correspond to sets of friends or colleagues; in computer networks communities may correspond to users in is peer-to-peer traffic or botnet farms [31]. Detecting this special substructure is also extremely useful in deriving social-based applications such as forwarding and routing strategies in communication networks [12, 21, 26], Sybil defense [28, 30], worm containment on cellular networks [26, 32], and sensor programming [27].

Newman–Girvan's modularity that measures the "strength" of partition of a network into modules (also called communities or clusters) [16] has rapidly become an essential element of many community detection methods. Modularity is by far the most used and best known quality function, particularly because of its success in many applications in social and biological networks [16]. One can search for community structure precisely by looking for the divisions of a network that have positive, and preferably large, values of the modularity. This is the main motivation for numerous optimization methods that find communities in the network via maximizing modularity as surveyed in [14]. Unfortunately, Brandes et al. [7] have shown that modularity maximization is an NP-hard problem. Thus, it is desirable to design algorithms maximizing modularity that provide lower bounds on the modularity values.

In contrast to the vast amount of work on maximizing modularity, the only known polynomial-time approach to find a good community structure with guarantees is due to Agarwal and Kempe [1] who rounded the fractional solution of a linear programming (LP). The value obtained by solving the LP gives an upper bound on the maximum achievable modularity. The main drawback of the approach is the large LP formulation that consumes both time and memory resources. As shown in their article, the approach can be used only on the networks of up to 235 nodes. In addition, no approximation ratio was proven for the proposed algorithms.

In this work, we study exact and approximation algorithms for maximizing modularity. In the first part, we investigate the complexity and approximability of the problem on tree graphs. Somewhat surprisingly, the problem is still NP-complete on trees. We then provide a polynomial time algorithm for uniform-weighted trees, a pseudopolynomial time algorithm, and a polynomial–time approximation scheme (PTAS) for trees with arbitrary weights. In the second part, we address the main drawback of the rounding LP approach by extending the *sparse metric* technique in [10]. We show both theoretically and experimentally that the new formulation substantially reduces the time and memory requirements without sacrificing the quality of the solution. The size of solved network instances rises from hundreds to several thousand nodes and the running time on the medium instances are accelerated from 10 to 150 times. In fact, the modularity values found by rounding LP are optimal for many network instances.

Related Work. A vast number of methods to find community structure is surveyed in [14]. Brandes et al. proves the NP-completeness for maximizing modularity, the first hardness result for this problem. Dasgupta et al. recently showed that maximizing modularity is APX-hard, i.e., it admits no PTAS [9].

Modularity has several known drawbacks. Fortunato and Barthelemy [13] have shown that the resolution limit (i.e., maximizing modularity methods fails to detect communities smaller than a scale), appears only when the network is substantially large [23]. Another drawback is that modularities highly degenerate energy landscape [17], which could lead to very different, yet equally high, modularity partitions. However, for small and medium networks of several thousand nodes, the method proposed by Blondel et al. [6] to optimize modularity is one of the best performing algorithms according to the Lancichinetti–Fortunato–Radicchi (LFR) benchmark [23]. Thus, our proposed method is applicable whenever high–quality solutions for small and medium networks are desired. Approximation algorithms for maximizing modularity are first studied in [11] for scale-free networks and in [9] for d-regular networks.

In [10], we introduce the sparse metric technique to construct compact formulations for the disruptor problem. The sparse metric helps reduce most redundant constraints and significantly shortens the solving time. In this article, we further refine the technique to reduce the number of constraints.

Organization. We present definitions and notations in Section 2. In Section 3, we present the first part of our results on maximizing modularity over tree graphs, including the NP-completeness, the polynomial-time algorithm for uniform-weighted trees, the pseudopolynomial time algorithm, and a PTAS for general weighted trees. We present, in Section 4 a sparse integer linear programming formulation for modularity maximization, by extending the sparse metric technique in [10]. The exact algorithm on trees is presented in Section 3. We show experimental results for the sparse formulation in Section 5 to illustrate the time efficiency of our formulations over the previous approach.

2. PRELIMINARIES

We consider a network represented as an undirected graph $G = (V, E)$ consisting of $n = |V|$ vertices and $m = |E|$ edges. The *adjacency matrix* of G is denoted by $\boldsymbol{A} = (A_{ij})$, where A_{ij} is the weight of edge (i, j) and $A_{ij} = 0$ if $(i, j) \notin E$. We also denote the (weighted) degree of vertex i, the total weights of edges incident at i, by $\mathsf{deg}(i)$ or, in short, d_i.

Community structure (CS) is a division of the vertices in V into a collection of disjoint subsets of vertices $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ that the union gives back to V. Each subset $C_i \subseteq V$ is called a *community* and we wish to have more edges connecting vertices in the same communities than edges that connect vertices in different communities. The *modularity* [25] of $\mathcal{C}$ is defined as

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2M} \right) \delta_{ij}, \tag{2.1}$$

where d_i and d_j are degree of nodes i and j, respectively, M is the total edge weights, and the element δ_{ij} of the membership matrix $\boldsymbol{\delta}$ is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same community} \\ 0, & \text{otherwise.} \end{cases}.$$

The modularity values can be either positive or negative and the higher (positive) modularity values indicate stronger community structure. The *maximizing modularity problem* asks to find a division that maximizes the modularity value.

We also define the modularity matrix $\boldsymbol{B}$ [25] with entries $B_{ij} = A_{ij} - \frac{d_i d_j}{2M}$. Then, the modularity can conveniently be written in the matrix form as

$$Q(\mathcal{C}) = \frac{1}{2M} \text{trace}(B\delta).$$

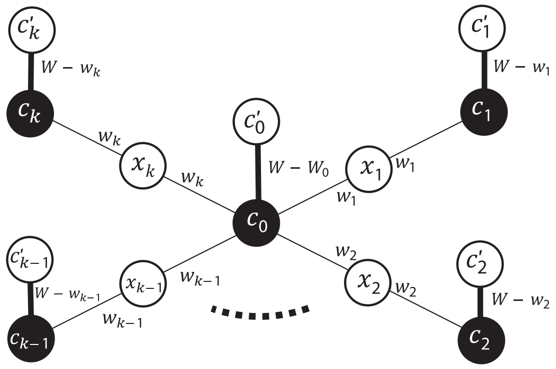

Figure 1 Gadget for NP-hardness of maximizing modularity on trees.

Alternatively, the modularity can be equivalently defined as

$$Q(\mathcal{C}) = \sum_{t=1}^{l} \left(\frac{E(C_t)}{M} - \frac{\mathsf{vol}(C_t)^2}{4M^2} \right), \tag{2.2}$$

where $E(C_t)$ is the total weight of edges whose endpoints are in C_t and for an $S \subseteq V$ we define $\mathsf{vol}(S) = \sum_{v \in S} d_v$, the volume of S.

3. COMPLEXITY AND APPROXIMATION ALGORITHMS ON TREES

We first show that maximizing modularity is even NP-hard on tree graphs. Then we propose a polynomial-time dynamic programming algorithm for the problem when the tree has uniform weights and its extension into a pseudopolynomial time algorithm to solve the problem on trees with arbitrary weights. The existence of the pseudopolynomial time algorithm implies that maximizing modularity on trees is a weakly NP-complete problem. Finally, we propose a PTAS for the problem that finds a CS with modularity at least $(1 - \epsilon)$ the maximum modularity within time $O(n^{1+1/\epsilon})$.

3.1. NP-Completeness

It has been proved in [7] that maximizing modularity is hard for unweighted graphs. We further prove that the decision version of maximizing modularity is still hard on weighted trees, one of the simplest graph classes. Our proof is much simpler than the first proof in [7], while implying the NP-hardness on a more restricted graph class.

Definition 3.1. **(Modularity on Trees)**. Given a tree $T = (V, E)$, weights $c(e) \in \mathbb{Z}^+$ for edges $e \in E$, and a number K, is there a community structure $\mathcal{C}$ of T, for which $Q(\mathcal{C}) \geq K$?

Our hardness result is based on a transformation from the following decision problem.

Definition 3.2. **(SUBSET-SUM)**. Given a set of k positive integers $w_1, w_2, \ldots, w_k$ and an integer S, does any nonempty subset sum to S?

We show that an instance $I = (\{w_1, w_2, \ldots, w_k\}; S)$ of SUBSET-SUM can be transformed into an instance $(T_I(V_I, E_I), c_I, K_I)$ of maximizing modularity on the tree T_I such that T_I has a community structure with modularity at least K_I if and only if there exists a subset of $\{w_1, w_2, \ldots, w_k\}$ that sums to S. Because SUBSET-SUM is an NP-hard problem [15], it follows that there is no polynomial-time algorithm to decide the maximizing modularity problem on trees, unless $P = NP$.

The reduction is as follows: Given an instance $I = (\{w_1, w_2, \ldots, w_k\}; S)$ of SUBSET-SUM, construct a tree $T_I = (V_I, E_I)$ consisting of $3k + 2$ vertices. For each integer $w_i \in I$, we introduce three vertices: a black vertex c_i and two white vertices c'_i and x_i. We also add a special pair: a black vertex c_0 and a white vertex c'_0. We connect each x_i to both c_i and c_0 with the weight w_i, and connect c'_i to c_i with weight $W - w_i$ for all $i = 1..n$, where $W = (2k + 1)\sum_i w_i$. The vertex c'_0 is connected to c_0 with weight $W - W_0$ where $W_0 = S + 1/2\sum_i w_i$. Before specifying parameters K_I, we characterize the important properties of any maximum modularity CS in a general graph $G = (V, E)$.

We summarize some known results of modularity maximization in the following lemma.

Lemma 3.3. *In a maximum modularity community structure of a graph $G = (V, E)$, the following properties hold.*

1. *There is no community with negative modularity.*
2. *Every nonisolated vertex is in the same community with at least one of its neighbors.*
3. *Each community induces a connected subgraph in G.*

The proof of (1) can be found in [12]. The proofs of (2) and (3) are similar to Lemma 3.3 and 3.4 in [7].

Lemma 3.4. *In a maximum modularity community structure of T_I, each community has exactly one black vertex.*

Proof. Consider a maximum modularity CS $\mathcal{C}$. The proof consists of two parts: 1) There is no community of $\mathcal{C}$ that has all white nodes; and (2) there is no community in $\mathcal{C}$ with more than one black vertex.

For the first part, observe that none of the adjacent vertices have the same color. By the second property of Lemma 3.3, if a community contains a vertex u, it also contains a neighbor of u whose the color is different from u's. Thus, every community must contain both black and white vertices.

We prove the second part by contradiction. Assume that a community $X \in \mathcal{C}$ has two black vertices. We show that $c_0 \in X$. Assume not, let c_i and c_j be two black vertices in X. Since $c_0 \notin X$, c_i, and c_j are disconnected within the subgraph induced by X in T_I, contradicting the third property in Lemma 3.3. Hence, we assume w.l.o.g. that X contains c_0 and c_1. We prove that dividing X into two communities $X_1 = \{c_1, c'_1\}$ and $X_2 = X \setminus X_1$ increases the modularity, which contradicts the optimality of $\mathcal{C}$. That is, to show

$$\Delta Q = -\frac{w_1}{M} + \frac{1}{4M^2}(\mathsf{vol}(X)^2 - \mathsf{vol}(X_1)^2 - \mathsf{vol}(X_2)^2) > 0, \tag{3.1}$$

where $M = (k+1)W + \sum_i w_i - W_0$ is the total weights of edges in T_I and for a given a subset $R \subseteq V$, $\mathsf{vol}(R)$ denotes the total weighted degree of the vertices in R.

Substitute $\mathsf{vol}(X) = \mathsf{vol}(X_1) + \mathsf{vol}(X_2)$ and simplify, we have

$$(3.1) \Leftrightarrow 4w_1 M < 2\mathsf{vol}(X_1)\mathsf{vol}(X_2). \tag{3.2}$$

Since $w_1 < \sum_i w_i = \frac{1}{2k+1}W$,

$$4w_1 M < \frac{2}{2k+1} W \left(k + 1 + \frac{1}{2k+1}\right) W < 2W^2. \tag{3.3}$$

On the right-hand side of (3.2):

$$\begin{aligned}
\mathsf{vol}(X_2) &> 2W - \sum_i w_i = 2W - \frac{1}{2k+1}W \\
\mathsf{vol}(X_1) &> \mathsf{deg}(c_0') + \mathsf{deg}(c_0) = 2W - 2W_0 + \sum_i w_i = 2W - 2S \\
&> 2W - 2\sum_i w_i = 2W - \frac{2}{2k+1}W.
\end{aligned}$$

When $k \geq 1$, we have

$$\begin{aligned}
2\mathsf{vol}(X_1)\mathsf{vol}(X_2) &> \left(2 - \frac{2}{2k+1}\right)\left(2 - \frac{1}{2k+1}\right) W^2 \\
&= \left(4 - \frac{6}{2k+1} + \frac{2}{(2k+1)^2}\right) W^2 \\
&> 2W^2.
\end{aligned} \tag{3.4}$$

It follows from (3.3) and (3.4) that (3.2) holds, i.e., $\Delta Q > 0$. This contradicts the maximum modularity of $\mathcal{C}$. Thus, each community in $\mathcal{C}$ must contain exactly one black vertex. □

Theorem 3.5. *Modularity maximization on trees is NP-complete.*

Proof. It is clear that maximizing modularity is in NP. To prove the NP-hardness, we reduce an instance $I = (\{w_1, w_2, \ldots, w_k\}; S)$ of SUBSET-SUM to an instance $(T_I(V_I, E_I), c_I, K_I)$ of maximizing modularity on the tree T_I as presented.

Consider a maximum modularity CS $\mathcal{C}$. From Lemma 3.4 and the third property of Lemma 3.3, x_i is in the same community with either c_0 or c_i (but not both). Let $\delta_i = 1$ if x_i is in the same community with c_0 and $\delta_i = 0$, otherwise. In addition, let $S_0 = \sum_{\delta_i=1} w_i$ and $W_S = \sum_{i=1}^k w_i = \frac{1}{2k+1}W$.

For $1 \leq i \leq k$, exactly one of the two edges (x_i, c_0) or (x_i, c_i) will have two endpoints in two different communities. This leads to an important property that the total weight of edges whose endpoints belong to different communities is exactly $\sum_{i=1}^k w_i$, we have:

$$Q(\mathcal{C}) = \left[1 - \frac{W_S}{M}\right] - \frac{1}{4M^2}\Big[(2W - 2W_0 + W_S + 2S_0)^2$$

$$+\sum_{\delta_i=0}(2W+w_i)^2+\sum_{\delta_i=1}(2W-w_i)^2\Bigg].$$

To maximize $Q(\mathcal{C})$, we need to minimize the second term that is

$$\frac{1}{4M^2}\Bigg[(2W-2W_0+W_S)^2+4S_0^2+4(2W-2W_0+W_S)S_0$$
$$+\sum_{\delta_i=0}(4W^2+4Ww_i+w_i^2)+\sum_{\delta_i=1}(4W^2-4Ww_i+w_i^2)\Bigg]$$
$$=\frac{1}{4M^2}\Bigg[(2W-2W_0+W_S)^2+4S_0^2+4(2W-2W_0+W_S)S_0$$
$$+4kW^2+4WW_S-8WS_0+\sum_{i=1}^{k}w_i^2\Bigg]$$
$$=\frac{1}{4M^2}\Bigg[\left((2W-2W_0+W_S)^2+4kW^2+4WW_S+\sum_{i=1}^{k}w_i^2\right)$$
$$+4\big(S_0^2-(2W_0-W_S)S_0\big)\Bigg].$$

For a fixed value of W_0, the above sum is minimized when $S_0^2-(2W_0-W_S)S_0$ is minimized at $S_0=\frac{2W_0-W_S}{2}=S$ (recall that we select $W_0=S+W_S/2$). Hence, if we choose

$$K_I=\left[1-\frac{W_S}{M}\right]-\frac{1}{4M^2}\left[(2W-2S)^2+4kW^2+4WW_S+\sum_{i=1}^{k}w_i^2)-4S^2\right],$$

there is a CS in T_I with modularity at least K_I if and only if there is a subset of w_i (corresponding to $\delta_i=1$) that sum to S. □

3.2. Polynomial-Time Algorithm for Uniform-Weighted Trees

We present a polynomial-time algorithm for finding a maximum modularity CS on trees with uniform edge weights. By characterizing the structure of a maximum modularity CS, we reduce maximizing modularity on trees to the following problem.

Definition 3.6. (k-MSSV problem). Given a tree $T=(V,E)$, find a set of k edges whose removal minimizes the sum-of-squares of component volumes in the residual forest.

The relationship between maximizing modularity and k-MSSV is presented in the following lemma.

Lemma 3.7. *Let $\mathcal{C}$ be a maximum modularity CS of $T=(V,E)$, k be the number of communities in $\mathcal{C}$, and F be the set of edges whose two endpoints belong to two different communities. Then the following properties hold.*

1. *For any two different communities, there is at most one edge that crosses between them.*
2. $|F| = k - 1$.
3. *F is an optimal solution for $(k-1)$-MSSV problem on $T = (V, E)$.*

Proof. By the first property in Lemma 3.3, each community induces a connected component. Thus, we shall use the terms *community and component* interchangeably in the rest of this section.

1. Assume that there are two different communities $C_1, C_2 \in \mathcal{C}$ and two different edges (u_1, v_1) and (u_2, v_2) that satisfy $u_1, u_2 \in C_1$ and $v_1, v_2 \in C_2$. By the first property in Lemma 3.3., there are paths between u_1 and u_2 within C_1 and between v_1 and v_2 within C_2. Those two paths together with the edges (u_1, v_1) and (u_2, v_2) form a cycle within the given tree T (contradiction).
2. Abstract each community in $\mathcal{C}$ into a single node, we obtain a new graph T_A with k abstract vertices. The set of edges in the new graph are identical to the edges in F. Since T is connected and cycle-free, T_A is also connected and cycle-free. Thus T_A is a tree with k vertices. It follows that T_A has $k-1$ edges and so does F.
3. By the second property, $Q(\mathcal{C}) = [1 - \frac{k-1}{|V|}] - \frac{1}{4|E|^2}\sum_{C_i \in \mathcal{C}} \mathsf{vol}(C_i)^2$. Thus, $Q(\mathcal{C})$ is maximized when $\sum_{C_i \in \mathcal{C}} \mathsf{vol}(C_i)^2$ is minimized and vice versa. Hence F is an optimal solution for the $(k-1)$-MSSV problem on T. □

Thus, a maximum modularity CS can be found by solving the k-MSSV problems with all k ranging from 0 to $|V|$. We introduce, following, a dynamic programming algorithm for maximizing modularity via solving the k-MSSV problem.

3.2.1. Dynamic Programming Algorithm. Given the tree $T = (V, E)$ with $|V| = n$, select a node $r \in V$ as the *root*. Denote by $T^u = (V^u, E^u)$ the subtree rooted at u in T with the set of vertices V^u and the set of edges E^u. Let $u_1, u_2, \dots, u_{b(u)}$ denote the children of u, where $t(u) = \mathsf{deg}(u)$ if $u = r$ and $t(u) = \mathsf{deg}(u) - 1$ if $u \neq r$. In our dynamic programming algorithm, we define the following functions:

- $F^u(k)$: The minimum sum-of-squares of component volumes in T^u when k edges are removed. Note that even after removing k edges, the component volumes are still measured as the sum of the vertex degrees in T, not T^u.
- $F^u(k, \nu)$: The minimum sum-of-squares of component volumes in T^u when k edges are removed and the component that contains u, called the *cap component*, has volume ν. In addition, if it is not possible to remove k edges to satisfy the two mentioned conditions, then $F^u(k, \nu) = \infty$.
- $F_i^u(k, \nu)$: The minimum sum-of-squares of component volumes in $T_i^u = (V_i^u, E_i^u)$, the partial subtree formed by $u, T^{u_1}, T^{u_2}, \dots, T^{u_i}$, when k edges are removed and the cap component has volume ν. As with previous function, if it is not possible to remove k edges to satisfy the two conditions, then $F_i^u(k, \nu) = \infty$.

The maximum modularity value is given at the root r by

$$\max_{1 \le k \le n-1, 1 \le \nu \le 2(n-1)} \left\{ \frac{k}{n} - \frac{F^r(k, \nu)}{4(n-1)^2} \right\}. \tag{3.5}$$

We compute $F^u(k, \nu)$ and $F_i^u(k, \nu)$ using the following recursions.

$$F^u(k) = \min_{d_u \le \nu \le \mathrm{vol}(T^u)} F^u(k,\nu) \quad \forall u \in V, k = 0..|E^u| \tag{3.6}$$

$$F^u(k,\nu) = F^u_{t(u)}(k,\nu) \qquad \forall u \in V, k = 0..|E^u|, \nu = d_u..\mathrm{vol}(T^u) \tag{3.7}$$

$$F_i^u(k,\nu) = \min \begin{cases} \min\limits_{0 \le l \le k-1} F^u_{i-1}(l,\nu) + F^{u_i}(k-l-1) & \text{(3.8.a)} \\ \min\limits_{0 \le l \le k, 0 \le \mu \le \nu} F^u_{i-1}(l,\mu) + F_i^{u_i}(k-l,\nu-\mu) + 2\mu(\nu-\mu) & \text{(3.8.b)} \end{cases}$$

$$\forall u \in V, k = 1..|E_i^u|, i = 1..t(u), \nu = 1..\mathrm{vol}(T^u) \tag{3.8}$$

The basis cases are as follows.

$$F_i^u(0,\nu) = \begin{cases} \mathrm{vol}(T_i^u)^2 & \text{if } \nu = \mathrm{vol}(T_i^u) \\ \infty & \text{otherwise} \end{cases} \tag{3.9}$$
$$\forall u \in V, k = 1..|E_i^u|, i = 1..t(u), \nu = 0..\mathrm{vol}(T^u)$$

$$F_0^u(k,\nu) = \begin{cases} d_u^2 & \text{if } \nu = d_u \\ \infty & \text{otherwise} \end{cases} \tag{3.10}$$
$$\forall u \in V, k = 0..|E_i^u|, \nu = 0..\mathrm{vol}(T^u)$$

$$F_i^u(k,\nu) = \infty \quad \forall u \in V, k = 0..|E_i^u|, i = 0..t(u), \nu < d_u \tag{3.11}$$

We explain the recursion from (3.6) to (3.8). In (3.6), we consider all possible ν, the volume of the cap component, and assign $F^u(k)$ the minimum values among $F^u(k,\nu)$. The definition of $F^u(k,\nu)$ is straightforward as shown in (3.7). Finally, we compute $F_i^u(k,\nu)$ in (3.8) as the minimum of the following two cases:

- If (u,u_i) is removed (3.8a): We can remove at most $k-1$ other edges. In addition, the cap component in T_i^u has the same volume ν with that of T^u_{i-1}. For $l = 0..k$, we take the sum of the optimal solution when removing l edges in T^u_{i-1} and the optimal solution when removing $k-l-1$ edges in T_{u_i} and take the minimum as a possible value for $T_i^u(k,\nu)$.
- If (u,u_i) is not removed (3.8b): We can remove l edges within T^u_{i-1} and $k-l$ removed edges within T^{u_i} for some $0 \le l \le k$. In addition, if the cap component volume of T^u_{i-1} is μ, then the cap component of T^{u_i} has volume $\nu-\mu$. The factor $2\mu(\nu-\mu) = \nu^2-\mu^2-(\nu-\mu)^2$ accounts for the increment of the total sum-of-squares volumes.

The running time of the dynamic algorithm is stated in the following lemma.

Lemma 3.8. *The recursions from (3.6) to (3.8) can be computed in $O(n^5)$.*

Proof. There are at most n different sets of $F_i^u(\cdot)$ to compute. For each set $F_i^u(\cdot)$, there are at most $n \times 2n$ pairs of (k,ν) and the time to compute each $F_i^u(k,\nu)$ is at most $n \times 2n$ (3.8). Thus, the running time is bounded by $O(n^5)$. □

We summarize the dynamic programming algorithm to maximize modularity in Algorithm 1 as follows.

Algorithm 1. Dynamic Programming Algorithm for Uniform-weight Trees

```
 1. for $u \in V$ in postorder traversal from $r$ do
 2.    for $i = 0$ to $t(u)$ do
 3.       for $k = 0$ to $|E(u)|$ do
 4.          for $\nu = 0$ to $\mathsf{vol}(T^u)$ do
 5.             Compute $F_i^u(k,\nu)$, $F^u(k,\nu)$, and $F^u(k)$ using (3.6)–(3.11).
 6. $Q = 0$
 7. for $k = 0$ to $n$ do
 8.    for $\nu = 0$ to $\mathsf{vol}(T)$ do
 9.       $Q = \max\{Q, \frac{k}{n} - \frac{F^r(k,\nu)}{4(n-1)^2}\}$
10. return $Q$
```

Theorem 3.9. *Algorithm 1 finds the maximum modularity in uniformly weighted trees in $O(n^5)$.*

3.3. Pseudopolynomial Time Algorithm

The above dynamic programming algorithm can be generalized to work for trees with *nonuniform integral weights*. The recursion is similar to (3.6)–(3.8) with the differences in the bounds for k and ν.

$$F^u(k) = \min_{d_u \le \nu \le \mathsf{vol}(T^u)} F^u(k,\nu) \quad \forall u \in V, k = 0..\mathsf{vol}(T^u)/2 \tag{3.12}$$

$$F^u(k,\nu) = F^u_{t(u)}(k,\nu) \quad \forall u \in V, k = 0..W^u, \nu = 0..\mathsf{vol}(T^u) \tag{3.13}$$

$$F_i^u(k,\nu) = \min \begin{cases} \min\limits_{0 \le l \le k} F_{i-1}^u(l,\nu) + F^{u_i}(k-l) & (3.14.a) \\ \min\limits_{0 \le l \le k-1, 0 \le \mu \le \nu} F_{i-1}^u(l,\mu) & \\ \qquad + F_i^{u_i}(k - l - w(u,u_i), \nu - \mu) + 2\mu(\nu - \mu) & (3.14.b) \end{cases} \tag{3.14}$$

$$\forall u \in V, k = 0..\mathsf{vol}(T^u), i = 1..t(u), \nu = 0..\mathsf{vol}(T_i^u). \tag{3.15}$$

The time complexity of the algorithm is a function of $vol(T)$, which can be exponentially large in terms of the input size. Thus, the dynamic programming is a pseudopolynomial time algorithm with the time complexity stated in the following theorem.

Theorem 3.10. *The recursion (3.12)–(3.15) gives an $O(n^5 W^4)$ pseudopolynomial time algorithm for maximizing modularity on trees with integral weights, where W is the maximum edge weight.*

Corollary 3.11. *Modularity maximization on weighted trees is weakly NP-complete.*

3.4. Polynomial Time Approximation Scheme (PTAS)

Given an $\epsilon > 0$, we present an $O(n^{1/\epsilon+1})$ algorithm to find CS in T with modularity at least $(1-\epsilon)Q_{\mathsf{opt}}$, where Q_{opt} denotes the maximum modularity over all possible CS.

On the one hand, the second property in Lemma 3.7 states that a CS with k communities has exactly $k - 1$ edges whose endpoints are in different communities. Thus there is a one-on-one correspondence between a set of $k - 1$ edges in T and CS with exactly k communities: removing $k - 1$ edges yields k connected components/communities. On the other hand, the following lemma implies that CS with at most k communities approximates closely the maximum modularity.

Lemma 3.12. [9, 11] *Given a weighted graph $G = (V, E)$, denote by Q_k the maximum modularity of a CS in G with* at most *k communities and by $Q_{\text{opt}} = Q_n$. We have*

$$Q_k \geq \left(1 - \frac{1}{k}\right) Q_{opt}.$$

Thus we have the following PTAS for maximizing modularity on trees.

Algorithm 2. PTAS for Maximizing Modularity on Trees

```
1. Given $\epsilon > 0$, set $k = \lceil 1/\epsilon \rceil$.
2. $Q_k = 0, \mathcal{C}_k = \{V\}$
3. for each $X \subset E$ and $|X| < k$ do
4.     Find connected component $C_1, C_2, \dots, C_k$ in $T' = (V, E \setminus X)$.
5.     Let CS $\mathcal{J} = \{C_1, C_2, \dots, C_k\}$
6. if $Q(\mathcal{J}) > Q_k$ then
7.     $Q_k = Q(\mathcal{J})$
8.     $\mathcal{C}_k = \mathcal{J}$
9. Return $\mathcal{C}_k$
```

Theorem 3.13. *Algorithm 2 finds in $O(n^{1+1/\epsilon})$ time a CS with modularity value of at least $(1 - \epsilon)Q_{opt}$ on weighted trees.*

Proof. By Lemma 3.12, the modularity of the found CS will be at least $(1 - \epsilon)Q_{\text{opt}}$. In addition, for each of the n^{k-1} subsets, computing the modularity takes $O(n)$. Thus, the total time complexity is $O(n^{k-1+1}) = O(n^{1/\epsilon+1})$. □

While the existence of an FPTAS implies the existence of a pseudopolynomial time algorithm, the reverse is not necessarily true. It is an open question whether or not an FPTAS for maximizing modularity on trees exists.

4. LINEAR PROGRAMMING-BASED ALGORITHM

In this section, we first present the original linear program (LP) in [1]. Then we present, in Subsection 4.2, our compact formulations that contain only a small fraction of constraints found in the original LP.

4.1. The Linear Program and the Rounding

Let x_{ij} be an $0 - 1$ integer that indicates whether nodes i and j are in different communities, i.e., $x_{ij} = 1$ if i and j are in different communities, and $x_{ij} = 0$, otherwise.

Note that x_{ij} is equivalent to $1 - \delta_{ij}$ in the Definition (3.1) of modularity. The modularity maximization problem can be formulated as an integer linear programming (ILP) as shown from (4.2) to (4.5) [1].

$$\begin{aligned}
\text{maximize} \quad & \frac{1}{2M}\sum_{ij} B_{ij}(1 - x_{ij}) & & \quad (4.1)\\
\text{subject to} \quad & x_{ij} + x_{jk} - x_{ik} \geq 0, & \forall i < j < k & \quad (4.2)\\
& x_{ij} - x_{jk} + x_{ik} \geq 0, & \forall i < j < k & \quad (4.3)\\
& -x_{ij} + x_{jk} + x_{ik} \geq 0, & \forall i < j < k & \quad (4.4)\\
& x_{ij} \in \{0, 1\}, & i, j \in [1..n] & \quad (4.5)
\end{aligned}$$

For convenience, we denote this ILP by IP_{cl}. Constraints (4.2), (4.3), and (4.4) are well-known *triangle inequalities* that guarantee that the values of x_{ij} are consistent with each other. They imply the following transitivity: if i and j are in the same community and j and k are in the same community, then so are i and k. By definition, $x_{ii} = 0\ \forall i$ and can be removed from the ILP for simplification.

We also refer to the relaxation of the IP_{cl}, obtained by replacing the constraints $x_{ij} \in \{0, 1\}$ by $x_{ij} \in [0, 1]$, as LP_{cl}. If the optimal solution of this relaxation is an integral solution, which is very often the case [2], we have a partition with the maximum modularity. Otherwise, we resort to rounding the fractional solution and use the value of the objective as an upperbound that enables us to lower bound the gap between the rounded solution and the optimal integral solution.

Agarwal and Kempe [1] used a simple rounding algorithm, proposed by Charikar et al. [8] for the *correlation clustering* problem [4], to obtain the community structure from a fractional optimal solution. The values of x_{ij} are interpreted as a metric "distance" between vertices. The algorithm repeatedly groups all vertices that are close to a vertex into a community. The final community structures are then refined by a Kernighan–Lin [22] local search method.

The modularity maximization formulation can also be expressed as a clique partitioning problem [18]. In [18], the authors proposed a row generation technique to incrementally add the triangle inequalities constraints and solve the LP. In each iteration, the batch of about 150 constraints is added and the nontight constraints are identified and removed from the LP.

In [10], we first proposed the *sparse metric* technique for the disruptor problem. The advantage of our sparse LP over the row generation method is that the sparse metric technique excludes major "redundant" constraints even before solving the programming formulation. Because the set of nontight constraints are known a priori, the LP is solved only once and we do not have to examine the $O(n^3)$ constraints in each iteration. Nevertheless, the two techniques are orthogonal and can be applied in parallel to improve the running time. Regarding efforts to solve the IP exactly, cutting planes and polyhedral characteristics for the clique partitioning and other clustering problems can be found in [2, 19, 20] and the references therein.

4.2. Sparse Formulations for Modularity Maximization

Instead of using $3\binom{n}{3}$ triangle inequalities, we show that only a small subset of those are sufficient to obtain the same optimal solutions.

Our integer linear program, denoted by IP$_{sp}$, is as follows:

$$\text{maximize} \quad -\frac{1}{2M}\sum_{ij} B_{ij}x_{ij} \tag{4.6}$$

$$\text{subject to} \quad x_{ik} + x_{kj} \geq x_{ij} \qquad k \in K(i,j) \subseteq V \setminus \{i,j\} \tag{4.7}$$

$$x_{ij} \in \{0,1\}. \tag{4.8}$$

First, since $\sum_{ij} B_{ij} = \sum_{ij} A_{ij} - \frac{d_i d_j}{2M} = 2M - \frac{\sum_i d_i \sum_j d_j}{2M} = 0$, we simplify the objective to $-\frac{1}{2M}\sum_{ij} B_{ij}x_{ij}$. Second, different selections of $K(i,j)$ give us different formulations. For example, the IP$_{cl}$ can be obtained by choosing $K(i,j) = V \setminus \{i,j\}$. However, not all selections of $K(i,j)$ result in valid formulations for the maximizing modularity problem.

Notice that if we define a function $d(i,j) = x_{ij}$, then the function should satisfy all the following conditions of a pseudometric:

1. $d(i,j) \geq 0$ (nonnegativity),
2. $d(i,i) = 0$ (and possibly $d(i,j) = 0$ for some distinct values $i \neq j$),
3. $d(i,j) = d(j,i)$ (symmetry), and
4. $d(i,j) \leq d(i,k) + d(k,j)$ (transitivity).

Therefore, $K(i,j)$ must be selected so that if $x \in [0,1]^{\binom{n}{2}}$ is a feasible solution, then x must induce a pseudometric.

We will prove in the next section (Theorems 4.3 and 4.4) that if $K(i,j)$ is a *vertex cut* for two vertices i and j, then IP$_{sp}$ is a valid formulation for the maximizing modularity problem. Here, a *vertex cut* of nodes i and j is a set of vertices whose removal from the graph disconnects i and j (if i and j are adjacent, we assume the edge i and j is also removed). Additionally, we show that the corresponding LP relaxation, called LP$_{sp}$, will have the same strength as the LP$_{cl}$, i.e., they have the same optimal objective values. There are many ways to select $K(i,j)$. One way is to select $K(i,j)$ as the minimum cardinality set between the set of neighbors of i or j as we did in [10]. In this study, we selected $K(i,j)$ as the *minimum cardinality vertex cut* of i and j to *minimize* the number of constraints. Then the cardinality of $K(i,j)$ is known as the vertex connectivity of i and j and is denoted by $\kappa(i,j)$.

Lemma 4.1. *If $d_1 \leq d_2 \leq \ldots \leq d_n$ is the sorted (unweighted) degree sequence of the graph, then the number of constraints is upper bounded by the following quantities*

$$(1)\ \sum_{i=1}^{n}(i-1)d_i, \qquad (2)\ m(n-1),$$

where $m = |E|$ is the number of edges.

Proof. Since $|K(i,j)| \leq \min\{d_i,d_j\}$, the number of constraints is, at most, $\sum_{i<j}\min\{d_i,d_j\} = \sum_{i<j} d_i = \sum_{i=1}^{n}(i-1)d_i$. Also, since $\min\{d_i,d_j\} \leq 1/2(d_i+d_j)$, the number of constraints is upper bounded by

$$\sum_{i<j}\frac{1}{2}(d_i+d_j) = \frac{1}{2}\sum_{i=1}^{n}(n-1)d_i = m(n-1).$$

This completes the proof. □

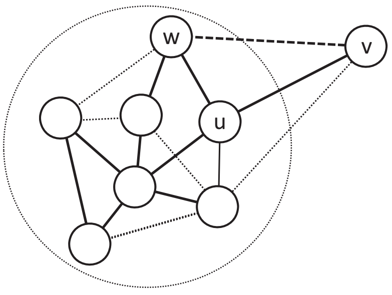

Figure 2 Clique-expanding process.

Corollary 4.2. *In sparse networks, where $m = O(n)$, LP_{sp} contains only $O(n^2)$ constraints.*

Thus, for sparse networks, our new formulations substantially reduce time and memory requirements. For most real-world network instances, where $n \approx m$, the number of constraints is effectively reduced from $\Theta(n^3)$ to $O(n^2)$. If we consider the time to solve a linear program to be cubic times the number of constraints, the total time complexity for sparse networks improves to $O(n^6)$ instead of $O(n^9)$ as in the original approach.

4.3. Validity and Strength of the Sparse Formulations

We show the equivalence between the sparse formulation and the complete formulation when $K(i, j)$ is selected as a *vertex cut* of i and j. We prove in Theorems 4.3 and 4.4 the following statements, respectively.

- IP$_{sp}$ and IP$_{cl}$ have the same set of optimal integral solutions.
- The optimal fractional solutions of LP$_{sp}$ and LP$_{cl}$ have the same objective values, i.e., they provide the same upper bound on the maximum possible modularity.

Hence, solving LP$_{sp}$ indeed gives us an optimal solution of LP$_{cl}$ within only a small fraction of time and the memory requirements of LP$_{cl}$.

Theorem 4.3. *Two integer programmings IP$_{sp}$ and IP$_{cl}$ have the same set of optimal solutions.*

Proof. We need to show that every optimal solution of IP$_{cl}$ is also a solution of IP$_{sp}$ and vice versa. In one direction, because the constraints in IP$_{sp}$ are a subset of constraints in IP$_{cl}$, every optimal solution of IP$_{cl}$ will also be a solution of IP$_{sp}$.

In the other direction, let x_{ij} be an optimal integral solution of IP$_{sp}$. We shall prove that x_{ij} must be a pseudometric that implies x_{ij} is also a feasible solution of IP$_{cl}$. For simplicity, we assume that the original graph $G = (V, E)$ has no isolated vertices [25]. Construct a graph $G_d = (V, E_d)$ in which there is an edge $(i, j) \in E_d$ for every $x_{ij} = 0$. Let $\mathcal{C}_d = \{C_d^1, C_d^2, \ldots, C_d^l\}$ be the set of connected components in G_d, where C_d^t is the set of vertices in the tth connected component.

We first prove an important property of the optimal community structure: each community must induce a connected subgraph in the network.

Claim: *Every connected component C_d^t induces a* connected *subgraph in the graph $G = (V, E)$.*

Proof. We prove by contradiction. Assume that the connected component C_d^t does not induce a connected subgraph in G. Hence, we can partition C_d^t into two subsets S and T so that there are no edges between S and T in G.

Construct a new solution x' from x by setting $x'_{ij} = 1$ for every pair $(i, j) \in S \times T$ and $x'_{ij} = x_{ij}$ otherwise. For every pair $(i, j) \in S \times T$, since $A_{ij} = 0$, we have $B_{ij} = A_{ij} - \frac{d_i d_j}{2M} < 0$. Hence, setting $x'_{ij} = 1\ \forall (i, j) \in S \times T$ can only increase the objective value. In fact, because there must be at least one pair $(i, j) \in S \times T$ with $x_{ij} = 0$ (or else C_d^t is not a connected component in G_d), doing so will strictly increase the objective. Moreover, we can verify that x' also satisfies all constraints of IP$_{sp}$. Thus, we have a feasible solution with a higher objective than the optimal solution x (contradiction). □

To prove that x_{ij} is a pseudometric, we prove an equivalent statement that if vertices i and j belong to the same component C_d^t, then the distance $x_{ij} = 0$. We prove by repeatedly growing a clique inside C_d^t. At each step, every pair of vertices in the clique are proven to have distance 0. Then, we add one more vertex to the clique and prove that the new vertex is also of distance zero from other vertices in the clique (see Figure 2).

Formally, we prove by induction that for all $p \leq |C_d^t|$, there exists in C_d^t a clique Ψ of size p satisfying simultaneously the following conditions

1. $x_{ij} = 0\ \forall (i, j) \in \Psi$, and
2. the subgraph induced by Ψ in G is connected.

The basis. For $p = 1$, select an arbitrary vertex in C_d^t. The previous two conditions hold trivially.

The inductive step. Assume that we have a clique $\Psi \subset C_d^t$ satisfying the two conditions. If $\Psi = C_d^t$, then we have completed the proof. Otherwise, there exist vertex $u \in \Psi$ and vertex $v \in C_d^t \setminus \Psi$ so that $(u, v) \in E(G)$ and $x_{uv} = 0$. The existence of such an edge (u, v) can be proven by contradiction. Assume not, then we can increase the distance of all pairs in $\Psi \times (C_d^t \setminus \Psi)$ from 0 to 1 to increase the objective value while not violating any constraints. This would imply that Ψ is disconnected from the rest of C_d^t, which is contradictory to the fact that C_d^t induces a connected subgraph. Now, consider an arbitrary neighbor w of u in the subgraph induced by Ψ. Because u is a common neighbor of w and v, we have $u \in K(w, v)$, i.e., the constraint $x_{wu} + x_{uv} \geq x_{wv}$ must be in IP$_{sp}$. Thus, we have $x_{wv} = 0$ since both $x_{wu} = 0$ $(w, u \in \Psi)$ and $x_{uv} = 0$.

We have proven that for every neighbor w of u, $x_{wv} = 0$. Similarly, for all neighbor w' of w, $x_{vw'} = 0$, and so on. Because the clique induces a connected subgraph in G, eventually $\forall u' \in \Psi$, we have $x_{vu'} = 0$. That is, we can extend the clique Ψ to include v and obtain a clique of size $p + 1$ that satisfies the two conditions. This completes the proof of Theorem 4.3. □

Theorem 4.4. *LP$_{sp}$ and LP$_{cl}$ share the set of optimal solutions that are extreme points.*

Proof. We need to show that every *fractional optimal* solution of LP$_{cl}$ is also a fractional solution of LP$_{sp}$ and vice versa. Because the integrality constraints have been dropped in both LP relaxations, we need a different approach to the proof in Theorem 4.3.3.

First, every fractional optimal solution of LP$_{cl}$ is also a fractional solution of LP$_{sp}$. For the other direction, let x be a fractional optimal solution of LP$_{sp}$; we shall prove that x is also a feasible solution of LP$_{cl}$.

Associate a weight $w_{ij} = x_{ij}$ for each edge $(i, j) \in E(G)$ (other edges are assigned weights ∞). Let x'_{ij} be the shortest distance between two vertices i and j within $G = (V, E)$ given the new edge weights. We shall prove the following statements:

1. $x'_{ij} = \min_{k=1}^{n}\{x'_{ik} + x'_{kj}\}$,
2. $x'_{ij} \geq x_{ij}$ for all i, j and $x'_{ij} = x_{ij}\ \forall (i, j) \in E$.

The first statement is obvious by the definition of x'_{ij}. We prove the second statement by contradiction. Assume that there exist i and j such that $x'_{ij} < x_{ij}$. Let $u_0 = i, u_1, \ldots, u_l = j$ be the vertices on the shortest path between i and j. We also assume that among pairs of vertices (i, j) satisfying $x'_{ij} < x_{ij}$, we select the pair with minimum value of l, the number of edges on the shortest path.

Because $K(i, j)$ is a vertex cut of i and j, there must be $0 < k < l$ such that $u_k \in K(i, j)$, i.e., the constraint $x_{iu_k} + x_{u_kj} \geq x_{ij} > x'_{ij}$ appears in the LP$_{sp}$. From the suboptimality of the shortest path, we have $x'_{iu_k} = x_{u_0u_1} + \ldots + x_{u_{k-1}u_k}$ and $x'_{u_kj} = x_{u_ku_{k+1}} + \ldots + x_{u_{l-1}u_l}$ and $x'_{ij} = x'_{iu_k} + x'_{u_kj}$. Therefore,

$$x_{iu_k} + x_{u_kj} \geq x_{ij} > x'_{ij} = x'_{iu_k} + x'_{u_kj},$$

which is either $x_{iu_k} > x'_{iu_k}$ or $x_{u_kj} > x'_{u_kj}$. However, the lengths of the shortest paths between i and u_k and between u_k and j are strictly less than l. We obtain the contradiction to the minimal selection of l and complete the proof of the second statement.

The two statements imply that x'_{ij} is a pseudometric. However, x'_{ij} may be no longer be upper bounded by one. Thus, we define $x^*_{ij} = \min\{x'_{ij}, 1\}$ which satisfies the following properties:

- $x^*_{ij} \geq x_{ij}\ \forall i, j$ (by definition),
- $x^*_{ij} = x'_{ij} = x_{ij}\ \forall (i, j) \in E$, and
- $x^*_{ik} + x^*_{kj} \geq \min\{x'_{ik} + x'_{kj}, 1\} \geq \min\{x'_{ij}, 1\} = x^*_{ij}$.

That is, x^* is also a pseudometric.

Now, if $x_{ij} = x^*_{ij}$ for all i, j, then x satisfies all triangle inequalities in LP$_{cl}$ and we yield the proof. Otherwise, assume that $x_{ij} < x^*_{ij}$ for some pair (i, j). We show that x^* is a feasible solution of LP$_{sp}$ with a greater objective value that contradicts the hypothesis that x is an optimal solution. Indeed, for all edges $(i, j) \notin E(G)$, $x_{ij} = x^*_{ij}$, and for $(i, j) \notin E$, we have $(B_{ij} < 0) \wedge (x^*_{ij} \geq x_{ij})$. Hence, the objective $-\frac{1}{2M}\sum_{ij} B_{ij}x^*_{ij} > -\frac{1}{2M}\sum_{ij} B_{ij}x_{ij}$ (contradiction). □

5. COMPUTATIONAL EXPERIMENTS

We compare the running time and the number of constraints of our sparse metric formulations and the original LP in [1]. In addition, we include in the comparison the

Problem ID	Name	Vertices (n)	Edges (m)
1	Zachary's karate club	34	78
2	Dolphin's social network	62	159
3	Les Miserables	77	254
4	Books about US politics	105	441
5	American College Football	115	613
6	US Airport 97	332	2126
7	Electronic Circuit (s838)	512	819
8	Scientific Collaboration	1589	2742

Table I Order and size of network instances.

modularity values of the most popular algorithms in the literature [1, 16, 25]. Also, we include the state of the art, the Blondel method, [6]. Because the/method Blondel is a randomized algorithm, we repeat the algorithm 10 times and report the best modularity value found. The optimal modularity values are reported in [2] except for the largest test case in which we use the GUROBI built-in, branch-and-cut algorithm to find the optimal integral solution.

We perform the experiments on the standard datasets for community structure identification [1,2], consisting of real-world networks. The datasets' names together with their sizes are listed in Table I. The largest network consists of 1580 vertices and 2742 edges. All references on the datasets can be found in [1] and [2]. The LP solver is GUROBI 4.5, running on a PC computer with Intel 2.93 Ghz processor and 12 GB of RAM.

Because the same rounding procedures are applied on the optimal fractional solutions, both LP$_{cl}$ and LP$_{sp}$ yield the same modularity values. However, LP$_{sp}$ can run on much larger network instances. The modularity of the rounding LP algorithms and other published methods are shown in Table II. The rounding LP algorithm can find optimal solutions (or within 0.1% of the optimal solutions) in all cases. Althought getting optimal modularity values with exact algorithms is costprohibitive, rounding fractional solutions of our LP$_{sp}$ takes less than 2 minutes for moderate-size networks.

ID	n	GN	EIG	Blondel	VP	LP$_{\mathrm{cl}}$	LP$_{\mathrm{sp}}$	OPT
1	34	0.401	0.419	0.420	0.420	0.420	0.420	0.420
2	62	0.520	—	0.524	0.526	0.529	0.529	0.529
3	77	0.540	—	0.560	0.560	0.560	0.560	0.560
4	105	—	0.526	0.527	0.527	0.527	0.527	0.527
5	115	0.601	—	0.605	0.605	0.605	0.605	0.605
6	332	—	—	—	—	—	0.368	0.368
7	512	—	—	0.796	—	—	0.819	0.819
8	1589	—	—	0.955	—	—	0.955	0.955

Table II The modularity obtained by previous published methods GN [16], EIG [25], Blondel (aka Louvain) [6], VP [1], LP$_{\mathrm{cl}}$ (complete LP) [1], our sparse formulation LP$_{\mathrm{sp}}$ and the optimal modularity values OPT [2].

ID	n	Constraint⟨C⟩	Constraint⟨S⟩	Time⟨C⟩	Time⟨S⟩
1	34	17,952	1,441	0.21	0.02
2	62	113,460	5,743	3.85	0.11
3	77	219,450	6,415	13.43	0.08
4	105	562,380	30,236	60.40	1.76
5	115	740,715	66,452	106.27	13.98
6	332	18,297,018	226,523	—	197.03
7	512	66,716,160	294,020	—	53.18
8	1589	2,002,263,942	159,423	—	2.94

Table III Number of constraints and running time in seconds of the formulations LP_{cl} and LP_{sp}. ⟨C⟩ stands for *complete* and ⟨S⟩ stands for *sparse*.

Note that only our rounding LP method can work on the test case 6, where the tested network is *directed*. The reason is that popular modularity optimization methods such as GN, EIG, and Blondel cannot work with directed networks; and the previous LP formulation [1] is too large to fit into the memory.

Finally, we compare the number of constraints of the LP formulation used in [1] and our new formulation (LP_{sp}) in Table III. Our new formulation contains substantially fewer constraints, thus, it can be solved more effectively. The old LP formulation cannot be solved within the time allowance (10,000 seconds) and the memory availability (12 GB) in cases of the network instances 6 to 8. The largest instance of 1589 nodes is solved surprisingly fast, taking under 3 seconds. The reason is due to the presence of leaves (nodes of degree one) and other special motifs that can be efficiently preprocessed with the reduction techniques in [3]. The size of solved network instances rises from hundreds to several thousand nodes, whereas the running time on the medium instances are accelerated from 10 to 150 times. Thus, our new formulation substantially reduces the time and memory requirements both theoretically and experimentally without any trade-off on the solution quality.

ACKNOWLEDGMENTS

We would like to thank the reviewers for very constructive and detailed comments, which improved the presentation of this work.

FUNDING

This work is partially supported by the DTRA YIP grant number HDTRA1-09-1-0061 and the NSF CAREER Award number 0953284.

REFERENCES

[1] G. Agarwal and D. Kempe. "Modularity-Maximizing Graph Communities via Mathematical Programming." *Eur. Phys. J. B* 66:3 (2008), 409–418.

[2] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti. "Column Generation Algorithms for Exact Modularity Maximization in Networks." *Phys. Rev. E* 82:4 (2010), 046112.

[3] A. Arenas, J. Duch, A. Fernandez, and S. Gomez. "Size Reduction of Complex Networks Preserving Modularity." *New J. Phys.* 9 (2007), 176.

[4] N. Bansal, A. Blum, and S. Chawla.“Correlation Clustering.” *Proceedings of the IEEE FOCS* 2002, 0:238. IEEE, 2002.

[5] A. Barabasi, R. Albert, and H. Jeong. “Scale-Free Characteristics of Random Networks: The Topology of the World-Wide Web.” *Physica A*, 281 (2000), 69–77.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. “Fast Unfolding of Communities in Large Networks.” *J.Stat. Mech. Theory* 2008:10 (2008), P10008.

[7] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. “On Modularity Clustering.” *IEEE Trans. Knowl. Data Eng*. 20:2 (2008), 172–188.

[8] M. Charikar and A. Wirth. “Maximizing Quadratic Programs: Extending Grothendieck’s Inequality.” In *Proceedings of the IEEE FOCS*, pp. 54–60. IEEE Computer Society, 2004.

[9] B. Dasgupta and D. Desai. “On the Complexity of Newman’s Community Finding Approach for Biological and Social Networks.” *J. Comput. Syst. Sci*. 79:1 (2013), 50–67.

[10] T. N. Dinh and M.T. Thai. “Precise Structural Vulnerability Assessment via Mathematical Programming.” In *Proceedings of the IEEE MILCOM 2011*, pp. 1351–1356. IEEE Computer Society, 2011.

[11] T. N. Dinh and M. T. Thai. “Finding Community Structure with Performance Guarantees in Scale-Free Networks.” In *Proceedings of the IEEE Social-Com/PASSAT*, pp. 888–891. IEEE, 2011.

[12] T. N. Dinh, Y. Xuan, and M. T. Thai. “Towards Social-Aware Routing in Dynamic Communication Networks. In *Proceedings of IEEE IPCCC*, pp.161–168. IEEE Computer Society, 2009.

[13] S. Fortunato and M. Barthelemy. “Resolution Limit in Community Detection. In *Proceedings of the National Academy of Sciences*, 104:1 (2007), 36–41.

[14] S. Fortunato and C. Castellano. Community Structure in Graphs. In *Encyclopedia of Complexity and Systems Science*. Berlin: Springer, 2008.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[16] M. Girvan and M. E. Newman. “Community Structure in Social and Biological Networks.” *Proceedings of the National Academy of Science* 99:12 (2002), 7821–7826.

[17] B. H. Good, Y.-A. de Montjoye, and A. Clauset. “Performance of Modularity Maximization in Practical Contexts.” *Phys. Rev. E*:81 (2010), 046–106.

[18] M. Grötschel and Y. Wakabayashi. “A Cutting Plane Algorithm for a Clustering Problem.” *Math. Program*. 45:1 (1989), 59–96.

[19] M. Grötschel and Y. Wakabayashi. “Facets of the Clique Partitioning Polytope.” *Math. Program*. 47:1 (1990), 367–387.

[20] P. Hansen and B. Jaumard. “Cluster Analysis and Mathematical Programming.” *Math. Program.* 79 (1997), 191–215.

[21] P. Hui, J. Crowcroft, and E. Yoneki. “Bubble Rap: Social-Based Forwarding in Delay-Tolerant Networks.” *IEEE Trans. on Mobile Comp*. 10 (2011), 1576–1589.

[22] B. W. Kemighan and S. Lin. “An Efficient Heuristic Procedure for Partitioning Graphs.” *J. of Classif.* 49:2 (1970), 291–307.

[23] A. Lancichinetti and S, Fortunato. “Community Detection Algorithms: A Comparative Analysis.” *Phys. Rev. E* 80 (2009), 056117.

[24] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. “Network Motifs: Simple Building Blocks of Complex Networks.” *Science* 298:5594 (2002), 763–764.

[25] M. E. J. Newman. “Modularity and Community Structure in Networks.” *Proceedings of the National Academy of Sciences* 103 (2006), 8577–8582.

[26] N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai. “Adaptive Algorithms for Detecting Community Structure in Dynamic Social Networks.” In *Proceedings of the IEEE INFOCOM 2011*, pp. 2282–2290. IEEE Computer Society, 2011.

[27] B. Pásztor, L. Mottola, C. Mascolo, G. P. Picco, S. Ellwood, and D. Macdonald. “Selective Reprogramming of Mobile Sensor Networks Through Social Community Detection.” *In Pro-*

ceedings of EWSN, Vol. 5970, pp. 178–193. Berlin, Heidelberg: Springer, 2010.

[28] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. “An Analysis of Social Network-Based Sybil Defenses.” In *Proceedings of the ACM SIGCOMM 2010*, pp. 363–374, New York, NY, USA: ACM, 2010.

[29] D. J. Watts and S. H. Strogatz. “Collective Dynamics of ’Small-World’ Networks.” *Nature* 393 (1998), 440–442.

[30] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. “Sybilguard: Defending Against Sybil Attacks via Social Networks. In *Proceedings of the ACM SIGCOMM 2006*, pp. 267–278, New York, NY, USA: ACM, 2006.

[31] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. “Botgraph: Large Scale Spamming Botnet Detection.” In *Proceedings of the USENIX NSDI 2009*, pp. 321–334. USENIX Association, 2009.

[32] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. “A Social Network Based Patching Scheme for Worm Containment in Cellular Networks.” In *Proceedings of the IEEE INFOCOM 2009*, pp. 1476–1484. IEEE Computer Society, 2009.